



UM0400

Ameba-D User Manual



Realtek Semiconductor Corp.

No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan

Tel.: +886-3-578-0211. Fax: +886-3-577-6047

www.realtek.com

COPYRIGHT

©2020 Realtek Semiconductor Corp. All rights reserved. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means without the written permission of Realtek Semiconductor Corp.

DISCLAIMER

Please Read Carefully:

Realtek Semiconductor Corp. ("Realtek") reference designs for WLAN products are solely designed for Customers ("Customer(s)") to use in the development of systems that incorporate Realtek products. Realtek's reference designs are provided "as is" without any warranties of any kind, and for reference only, Customers remain responsible for the systems that they design, test, and evaluate. Realtek may make improvements and/or changes to this reference designs at any time and at its sole discretion. With respect to the document, software, information, materials, services, and any improvements and/or changes thereto provided by Realtek, Realtek disclaims all warranties, whether express, implied or otherwise, including, without limitation, implied warranties of merchantability or fitness for a particular purpose, and non-infringement.

Realtek's reference designs are solely intended to assist Customers who are developing systems that incorporate Realtek products. While Realtek does update this information periodically, please contact Realtek for the latest updated reference designs. Realtek reserves the right to make corrections, enhancements, improvements and other changes to Realtek reference designs and other items.

Realtek's provision of reference designs and any other technical, applications or design advice, simulations, quality characterization, reliability data or other information or services does not expand or otherwise alter Realtek's applicable published warranties or warranty disclaimers for Realtek products, and no additional obligations or liabilities arise from Realtek providing such Realtek reference designs or other items.

Customer understands and agrees that Customer remains responsible for using its independent analysis, evaluation and judgment in designing Customer's systems and products, and has full and exclusive responsibility to assure the safety of its products and compliance of its products (and of all Realtek products used in or for such Customer's products) with all applicable regulations, laws and other applicable requirements. Customer represents that, with respect to its applications, it has all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. Customer agrees that prior to using or distributing any systems that include Realtek's reference designs, Customer will thoroughly test such systems and the functionality of such Realtek products used in such systems. Customer may not use any Realtek's reference designs in life-critical medical equipment unless authorized officers of the parties have executed a special contract specifically governing such use. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death. Such equipment includes, without limitation, all medical devices identified by the U.S. FDA as Class III devices and equivalent classifications outside the U.S.

Customers are authorized to use, copy and modify any individual Realtek's reference designs only in connection with the development of end products that include the Realtek's products. However, no other license, express or implied, by estoppel or otherwise to any other Realtek's intellectual property right, and no license to any technology or intellectual property right of Realtek or any third party is granted herein, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Realtek products, reference designs or services are used. Information published by Realtek regarding third party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of the Realtek's reference designs or other items may require a license from a third party under the patents or other intellectual property of the third party, or a license from Realtek under the patents or other intellectual property of Realtek.

Realtek's reference designs and other items described above are provided "as is" and with all faults. Realtek disclaims all other warranties or representations, express or implied, regarding the Realtek reference designs or use of the Realtek, including but not limited to accuracy or completeness, title, any epidemic failure warranty and implied warranties of merchantability, fitness for a particular purpose, and non-infringement of any third party intellectual property rights.

Realtek shall not be liable for and shall not defend or indemnify Customers against any claim, including but not limited to any infringement claim that relates to or is based on any combination of products as described in Realtek's reference designs or otherwise. In no event shall Realtek be liable for any actual, direct, special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of Realtek's reference designs or use of Realtek's reference designs, and regardless of whether Realtek has been advised of the possibility of such damages.

Customers will fully indemnify Realtek and its representatives against any damages, costs, losses, and/or liabilities arising out of Customer's non-compliance with the terms and provisions of this Notice.

TRADEMARKS

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

USING THIS DOCUMENT

Though every effort has been made to ensure that this document is current and accurate, more information may have become available subsequent to the production of this guide.

Contents

Contents.....	4
List of Tables.....	12
List of Figures	14
Conventions	19
1 Product Overview	20
1.1 General Description	20
1.2 System Architecture	20
2 Memory Organization.....	22
2.1 Introduction.....	22
2.2 KM4 Memory.....	22
2.2.1 Memory Map and Register Boundary Addresses	22
2.2.2 Embedded SRAM	23
2.2.3 Extension SRAM.....	23
2.3 KM0 Memory.....	24
2.3.1 Memory Map and Register Boundary Addresses	24
2.3.2 Embedded SRAM	24
2.4 Retention SRAM	24
2.5 SPI Flash Memory	24
2.6 PSRAM	25
3 Memory Protection Unit (MPU).....	26
3.1 Register Map	26
3.2 Register Field Description	26
3.2.1 MPU_TYPE	26
3.2.2 MPU_CTRL	27
3.2.3 MPU_RNR	28
3.2.4 MPU_RBAR	28
3.2.5 MPU_RLAR.....	29
3.2.6 MPU_RBAR_A<n>.....	30
3.2.7 MPU_RLAR_A<n>	30
3.2.8 MPU_MAIRO	31
3.2.9 MPU_MAIR1	32
3.3 Memory Attribute Indirection Register Attributes (MAIR_ATTR)	32
3.3.1 Outer.....	33
3.3.2 Inner	33
4 Nested Vectored Interrupt Controller (NVIC).....	34
4.1 Features	34
4.2 NVIC Diagram.....	34
4.3 NVIC Table	34
4.4 NVIC Register Description.....	36
4.4.1 ISER0 and ISER1	37
4.4.2 ICER0 and ICER1	38

4.4.3	ISPRO and ISPR1.....	38
4.4.4	ICPRO and ICPR1.....	38
4.4.5	IABRO and IABR1.....	38
4.4.6	IPRO ~ IPR14.....	38
4.4.7	STIR.....	38
5	CPU System Tick (SysTick) Timer.....	40
5.1	Features.....	40
5.2	Functional Description.....	40
5.3	Register Description.....	40
5.3.1	SYST_CSR.....	40
5.3.2	SYST_RVR.....	41
5.3.3	SYST_CVR.....	41
5.3.4	SYST_CALIB.....	42
6	Pad Control and Pinmux.....	43
6.1	Features.....	43
6.2	Functional Description.....	43
6.2.1	Pad Types.....	43
6.2.2	Pad Pull Resistor Control.....	44
6.2.3	Pad Schmitt Trigger.....	45
6.2.4	Pad Driving Strength.....	45
6.2.5	Pad Shutdown.....	45
6.2.6	I ² C Open-drain Mode.....	45
6.2.7	Audio Pad.....	45
6.3	Pin Multiplexing Function.....	47
6.4	Register – PADCTRL.....	48
7	Inter Processor Communication (IPC).....	49
7.1	Features.....	49
7.2	Functional Description.....	49
7.2.1	Architecture Block Diagram.....	49
7.2.2	Core-to-Core Interrupt.....	49
7.2.3	Hardware Semaphore.....	50
7.3	IPC Registers.....	51
7.3.1	IPCx_IER.....	51
7.3.2	IPCx_IDR.....	52
7.3.3	IPCx_IRR.....	52
7.3.4	IPCx_ICR.....	52
7.3.5	IPC0_CPUID.....	53
7.3.6	IPCx_ISR.....	53
7.3.7	IPC0_SEM.....	53
7.3.8	IPCx_IER_R.....	54
7.3.9	IPC_USR.....	55
8	General Purpose Input/Output (GPIO).....	56
8.1	Introduction.....	56
8.1.1	General Product Description.....	56
8.1.2	Features.....	56
8.2	Functional Description.....	56
8.2.1	Data and Control Flow.....	57
8.2.2	Interrupts.....	59
8.3	Registers.....	67

8.3.1	Bus Interface.....	67
8.3.2	Register Memory Map.....	68
8.3.3	Register and Field Descriptions.....	69
8.4	Programming the GPIO.....	83
8.4.1	Software Registers.....	83
8.4.2	Programming Considerations.....	83
9	Direct Memory Access Controller (DMAC)	84
9.1	Product Overview	84
9.1.1	General Product Description.....	84
9.1.2	Basic Definitions	86
9.1.3	Features.....	88
9.2	Functional Description.....	90
9.2.1	Setup/Operation of DMA Transfers.....	90
9.2.2	Block Flow Controller and Transfer Type	90
9.2.3	Handshaking Interface	91
9.2.4	Basic Interface Definitions	91
9.2.5	Memory Peripherals	92
9.2.6	Handshaking Interface – Peripheral is Not Flow Controller.....	92
9.2.7	Handshaking Interface – Peripheral Is Flow Controller	98
9.2.8	Setting up Transfers.....	100
9.2.9	Flow Control Configurations	119
9.2.10	Peripheral Burst Transaction Requests.....	120
9.2.11	Generating Requests for the AHB Master Bus Interface	124
9.2.12	Arbitration for AHB Master Interface	126
9.2.13	Scatter/Gather.....	127
9.2.14	Endianness.....	129
9.2.15	AHB Transfer Error Handling	130
9.3	Registers	130
9.3.1	Register Memory Map.....	130
9.3.2	Registers and Field Descriptions.....	135
9.4	Programming the DMAC.....	170
9.4.1	Register Access	170
9.4.2	Illegal Register Access.....	170
9.4.3	DMA Transfer Types	171
9.4.4	Programing Example	175
9.4.5	Programming a Channel.....	177
9.4.6	Disabling a Channel Prior to Transfer Completion.....	192
9.4.7	Defined-Length Burst Support on DMAC	193
10	General Timers.....	194
10.1	Basic Timer.....	194
10.1.1	Introduction.....	194
10.1.2	Features.....	194
10.1.3	Block Diagram	194
10.1.4	Functional Description.....	194
10.2	Pulse Mode Timer	195
10.2.1	Introduction.....	195
10.2.2	Features.....	195
10.2.3	Block Diagram	196
10.2.4	Functional Description.....	196
10.3	PWM Mode Timer.....	198
10.3.1	Introduction.....	198

10.3.2	Features.....	198
10.3.3	Block Diagram	199
10.3.4	Functional Description.....	199
10.4	Registers.....	206
10.4.1	TIM0/TIM1/TIM2/TIM3 Registers	206
10.4.2	TIM4 Registers.....	209
10.4.3	TIM5 Registers.....	213
10.5	Design Implementation.....	227
10.5.1	Introduction.....	227
10.5.2	Synchronous Data from Fast Clock to Slow Clock.....	228
10.6	Operation Flow	229
10.6.1	Upcounting Mode.....	229
10.6.2	Pulse Mode.....	229
10.6.3	PWM Mode.....	230
10.6.4	Input Capture Mode	231
11	Real-time Clock (RTC)	232
11.1	Product Overview	232
11.1.1	Introduction.....	232
11.1.2	Features.....	232
11.1.3	Block Diagram	232
11.1.4	RTC Clock Select Diagram.....	233
11.2	Functional Description	233
11.2.1	Clock and Prescaler.....	233
11.2.2	32K Auto-trigger Calibration Circuit	234
11.2.3	Programmable Alarm	235
11.2.4	Write Protection	235
11.2.5	Digital Calibration.....	235
11.2.6	Day Threshold Program.....	236
11.3	Registers.....	236
11.3.1	RTC Time Register (RTC_TR)	236
11.3.2	RTC Control Register (RTC_CR)	236
11.3.3	RTC Initialization and Status Register (RTC_ISR)	238
11.3.4	RTC Prescaler Register (RTC_PRER)	239
11.3.5	RTC Calibration Register (RTC_CALIBR)	239
11.3.6	RTC Alarm 1 Register Low (RTC_ALMR1L)	240
11.3.7	RTC Alarm 1 Register High (RTC_ALMR1H)	240
11.3.8	RTC Write Protection Register (RTC_WPR).....	241
11.3.9	RTC 32K Auto-calibration Register (RTC_CLKACALR).....	241
11.4	Operation Flow	242
11.4.1	Initialize the Calendar.....	242
11.4.2	Configure Alarm.....	242
11.4.3	Configure Calibration.....	242
11.4.4	Daylight Saving Time	243
12	Watchdog Timer (WDT).....	244
12.1	Introduction	244
12.2	Features	244
12.3	Registers.....	244
13	Inter-integrated Circuit (I²C) Interface	246
13.1	Product Introduction.....	246
13.2	Functional Description	246

13.2.1	Overview.....	246
13.2.2	I ² C Terminology	247
13.2.3	I ² C Behavior	248
13.2.4	I ² C Protocols.....	249
13.2.5	Tx FIFO Management and START, STOP and RESTART Generation	253
13.2.6	Multiple Master Arbitration	255
13.2.7	Clock Synchronization	256
13.2.8	Operation Modes.....	256
13.2.9	IC_CLK Frequency Configuration.....	258
13.2.10	Programmable SDA Hold Time.....	260
13.2.11	DMA Controller Interface.....	260
13.2.12	Low Power Mode	261
13.3	Registers.....	262
13.3.1	Register Memory Map.....	262
13.3.2	Registers and Field Descriptions.....	263
14	Universal Asynchronous Receiver/Transmitter (UART)	285
14.1	Introduction	285
14.1.1	Features.....	285
14.1.2	Block Diagram	285
14.2	Register	286
14.2.1	IER.....	287
14.2.2	IIR.....	287
14.2.3	LCR.....	288
14.2.4	MCR	289
14.2.5	LSR	290
14.2.6	MSR	290
14.2.7	SCR.....	291
14.2.8	STSR.....	292
14.2.9	RBR	292
14.2.10	THR.....	292
14.2.11	MISCR.....	293
14.2.12	IRDA_SIR_TX_PW_CTRL	294
14.2.13	IRDA_SIR_RX_PW_CTRL	294
14.2.14	BAUD_MON	294
14.2.15	DBG_UART	295
14.2.16	REG_RX_PATH_CTRL	295
14.2.17	REG_MON_BAUD_CTRL	296
14.2.18	REG_MON_BAUD_STS	296
14.2.19	REG_MON_CYC_NUM.....	296
14.2.20	REG_RX_BYTE_CNT.....	297
14.2.21	FCR	297
14.3	Design Implementation.....	298
14.3.1	Baud Rate Calculation	298
14.3.2	Clock Structure of Rx Path	299
14.3.3	Ird_a_sir_encoder/decoder	300
14.3.4	Auto-flow Control	301
14.3.5	Interrupt Control.....	301
14.3.6	DMA Flow Control	302
15	Infrared Radiation (IR).....	303
15.1	Overall Description	303
15.1.1	Introduction.....	303

15.1.2	Features.....	304
15.2	Architecture	304
15.2.1	Scaler.....	305
15.2.2	Glitch Filter	305
15.2.3	Interrupt	305
15.3	Registers.....	305
15.3.1	IR Clock Control Register.....	306
15.3.2	IR Tx Registers	306
15.3.3	IR Rx Registers	309
15.3.4	IR Version Register.....	313
15.4	IR Application Note	313
15.4.1	RCU Application.....	313
15.4.2	Receiver Application	314
16	Key-Scan.....	315
16.1	Overall Description	315
16.1.1	Application Scenario.....	315
16.1.2	Features.....	315
16.2	Functional Description	315
16.2.1	Block Diagram	315
16.2.2	Work Principle	316
16.2.3	FIFO Mechanism.....	319
16.2.4	Clock Configuration	319
16.2.5	Shadow Key Problem.....	320
16.3	Registers.....	322
16.3.1	KS_CLK_DIV	322
16.3.2	KS_TIM_CFG0	323
16.3.3	KS_TIM_CFG1	323
16.3.4	KS_CTRL	323
16.3.5	KS_FIFO_CFG	324
16.3.6	KS_COL_CFG	324
16.3.7	KS_ROW_CFG	325
16.3.8	KS_DATA_NUM	325
16.3.9	KS_DATA.....	326
16.3.10	KS_IMR.....	326
16.3.11	KS_ICR.....	327
16.3.12	KS_ISR	327
16.3.13	KS_ISR_RAW	328
16.3.14	KS_DUMMY.....	328
17	Audio Codec (AC)	330
17.1	Introduction	330
17.2	Diagram.....	330
17.3	Key Features.....	331
17.3.1	Analog Part.....	331
17.3.2	Digital Part.....	331
17.4	Specifications	332
17.4.1	DAC Path.....	332
17.4.2	ADC Path.....	333
17.5	Application and Implementation	334
17.5.1	Audio Output	334
17.5.2	Audio Input	335
17.6	Registers.....	338

17.6.1	Analog Part.....	338
17.6.2	Digital Part.....	342
17.6.3	DAC_EQ	361
17.6.4	ADC_EQ	371
18	Audio Codec Controller (ACC).....	382
18.1	Introduction	382
18.2	Features	382
18.3	Architecture	382
18.3.1	Block Diagram	382
18.3.2	Data Part	383
18.3.3	Control Part	388
18.3.4	ACC Clock.....	389
18.4	Registers.....	389
18.4.1	SPORT Control Registers.....	389
18.4.2	SI Control Registers.....	394
19	Serial Peripheral Interface (SPI).....	396
19.1	Product Overview	396
19.1.1	Block Diagram	396
19.1.2	Features.....	396
19.2	Functional Description	397
19.2.1	Overview.....	397
19.2.2	Transfer Modes.....	401
19.2.3	Operation Modes.....	401
19.2.4	DMA Controller Interface.....	405
19.3	Registers.....	405
19.3.1	Register Memory Map.....	405
19.3.2	Registers and Field Descriptions.....	407
20	Liquid Crystal Display Controller (LCDC)	422
20.1	Overall Description	422
20.1.1	Introduction.....	422
20.1.2	Features.....	422
20.1.3	LCD Application Scenario.....	422
20.2	Architecture	423
20.2.1	Block Diagram	423
20.2.2	MCU Interface	425
20.2.3	RGB Interface.....	428
20.2.4	LED Control	431
20.2.5	Pinmux.....	437
20.2.6	Supported Resolution	437
20.3	Registers.....	438
20.3.1	Global Control Registers.....	439
20.3.2	Interrupt and Status Registers.....	442
20.3.3	RGB Control Registers.....	445
20.3.4	MCU Control Registers	447
20.3.5	LED Control Registers	450
20.3.6	Image Control Registers	451
20.4	Programming the LCDC.....	452
20.4.1	RGB DMA Auto-mode	452
20.4.2	MCU DMA Trigger-mode	452
20.4.3	MCU I/O Mode	453

21	Quadrature Decoder (Q-Decoder)	454
21.1	Overall Description	454
21.1.1	Introduction	454
21.1.2	Features	454
21.1.3	Application Scenario	454
21.2	Architecture	455
21.2.1	Q-Decoder Block Diagram	455
21.2.2	Position Measurement	456
21.2.3	Velocity Measurement	460
21.3	Registers	461
21.3.1	Global Control Registers	461
21.3.2	Position Measurement Registers	464
21.3.3	Velocity Measurement Registers	466
21.3.4	Interrupt Registers	469
22	Inter-IC Sound (I²S)	473
22.1	Introduction	473
22.2	Features	473
22.3	Interface	473
22.4	Functional Description	474
22.4.1	Signal Lines	474
22.4.2	Operation Mode	475
22.4.3	Serial Data Standard	476
22.4.4	Clock Type	477
22.4.5	Memory Block	478
22.4.6	FIFO Allocation	479
22.5	Registers	482
22.5.1	Control Register (IS_CTL)	482
22.5.2	Tx Page Pointer Register (IS_TX_PAGE_PTR)	483
22.5.3	Rx Page Pointer Register (IS_RX_PAGE_PTR)	484
22.5.4	Page Size and Sample Rate Setting Register (IS_SETTING)	484
22.5.5	Tx Interrupt Enable Register (IS_TX_MASK_INT)	484
22.5.6	Tx Interrupt Status Register (IS_TX_STATUS_INT)	485
22.5.7	Rx Interrupt Enable Register (IS_RX_MASK_INT)	486
22.5.8	Rx Interrupt Status Register (IS_RX_STATUS_INT)	487
22.5.9	Tx Page Own Bit Register (IS_TX_PAGE_OWNx)	488
22.5.10	Rx Page Own Bit Register (IS_RX_PAGE_OWNx)	488
22.5.11	Version ID (IS_VERSION_ID)	488
	Abbreviations	491
	Revision History	495

List of Tables

Table 2-1 Address space main blocks	22
Table 2-2 KM4 register boundary addresses	22
Table 2-3 KM0 register boundary addresses	24
Table 3-1 MPU entries	26
Table 3-2 Register map of MPU	26
Table 4-1 NVIC table	35
Table 4-2 Memory map of NVIC	36
Table 4-3 Software trigger interrupt register	39
Table 5-1 Memory map of SysTick timer	40
Table 6-1 Pad types.....	44
Table 6-2 I ² C pad pull resistor control.....	44
Table 6-3 Schmitt trigger specification	45
Table 6-4 Normal pad driving strength configuration	45
Table 6-5 Using audio pins as normal GPIOs	46
Table 6-6 Audio LDO voltage setting.....	46
Table 6-7 Mute configuration	47
Table 7-1 IPC control registers.....	51
Table 8-1 Memory map of GPIO	68
Table 9-1 Transfer types and flow control combinations	90
Table 9-2 Hardware handshaking interface	94
Table 9-3 Hardware handshaking interface	99
Table 9-4 Parameters Used in Transfer Examples	100
Table 9-5 Parameters in transfer operation – Example 1	101
Table 9-6 Parameters in transfer operation – Example 4	106
Table 9-7 Parameters in transfer operation – Example 5	108
Table 9-8 Parameters in transfer operation – Example 6	109
Table 9-9 Parameters in transfer operation – Example 7	112
Table 9-10 Transfer parameters	115
Table 9-11 Parameters in transfer operation – Example 7, Case 2b.....	117
Table 9-12 Transmit watermark level – Case 1	121
Table 9-13 Transmit watermark level – Case 2	122
Table 9-14 Memory map of DMAC	130
Table 9-15 CTLx.SRC_MSIZ and DEST_MSIZ decoding	142
Table 9-16 CTLx.SRC_TR_WIDTH and CTLx.DST_TR_WIDTH decoding.....	142
Table 9-17 CTLx.TT_FC field decoding	142
Table 9-18 PROTCTL field to HPROT mapping	148
Table 9-19 Programming of transfer types and channel register update method	173
Table 10-1 Basic timers features.....	194
Table 10-2 Pulse timer features.....	195
Table 10-3 PWM timer features	198
Table 10-4 TIM0~TIM5 address table	206
Table 10-5 TIM0/TIM1/TIM2/TIM3 memory map	206
Table 10-6 TIM4 memory map	209
Table 10-7 TIM5 memory map	213
Table 10-8 Timers upcounting configuration flow	229
Table 10-9 Pulse mode 0 configuration flow	229
Table 10-10 Pulse mode 1 configuration flow	229
Table 10-11 PWM repeated mode configuration flow	230
Table 10-12 TIM5 one-pulse mode configuration flow	230

Table 10-13 TIM5 input capture mode configuration flow.....	231
Table 11-1 Example of calibrating resolution	235
Table 11-2 Register table of RTC.....	236
Table 11-3 Initialization calendar configure flow.....	242
Table 11-4 Alarm configure flow.....	242
Table 11-5 Calibration configure flow.....	242
Table 13-1 I ² C definition of bits in the first byte	250
Table 13-2 I ² C Memory map	262
Table 14-1 Register map of UART	286
Table 14-2 Supported baud rate of Tx path & Rx path	298
Table 14-3 Error rate of baud rate	298
Table 14-4 Signaling rate and pulse duration specifications	300
Table 14-5 Description of different interrupt	301
Table 15-1 Interrupts	305
Table 15-2 IR registers	305
Table 16-1 Key value assignment.....	319
Table 16-2 Register map of Key-Scan.....	322
Table 17-1 Analog key features	331
Table 17-2 Digital key features	331
Table 17-3 DAC path electrical characteristics	332
Table 17-4 ADC path electrical characteristics	333
Table 18-1 16-bit stereo data without lr_swap/byte_swap	383
Table 18-2 16-bit mono data without lr_swap/byte_swap	383
Table 18-3 16-bit stereo data with lr_swap	384
Table 18-4 16-bit mono data with lr_swap.....	384
Table 18-5 16-bit stereo data with byte_swap	384
Table 18-6 16-bit mono data with byte_swap.....	384
Table 18-7 24-bit stereo data without lr_swap/byte_swap	384
Table 18-8 24-bit mono data without lr_swap/byte_swap	385
Table 18-9 24-bit stereo data with lr_swap	385
Table 18-10 24-bit mono data with lr_swap.....	385
Table 18-11 24-bit stereo data with byte_swap	385
Table 18-12 24-bit mono data with byte_swap.....	385
Table 18-13 8-bit mono data without lr_swap/byte_swap	385
Table 18-14 SPORT main configuration	386
Table 18-15 SPORT control register layout	389
Table 18-16 SI control register layout.....	394
Table 19-1 Memory map of SPI.....	406
Table 19-2 DFS decode	408
Table 19-3 TFT decode.....	411
Table 19-4 RFT decode.....	411
Table 19-5 DMATDL decode value.....	418
Table 19-6 DMARDL decode value.....	419
Table 20-1 MCU interface type	427
Table 20-2 LCDC pinmux	437
Table 20-3 Memory map of LCDC	438
Table 20-4 Typical application scenario of LCDC	452
Table 21-1 Q-Decoder registers	461
Table 22-1 Clock configuration	478
Table 22-2 Memory map of I ² S	482

List of Figures

Fig 1-1 System architecture	20
Fig 4-1 NVIC diagram.....	34
Fig 6-1 Pad diagram	43
Fig 6-2 Circuit diagram of audio pad	46
Fig 6-3 Selecting an alternate function on Ameba-D	47
Fig 7-1 IPC system architecture.....	49
Fig 7-2 IPC interrupt request.....	50
Fig 7-3 IPC hardware semaphore mechanism	50
Fig 8-1 Block diagram	56
Fig 8-2 Control RTL block diagram	57
Fig 8-3 Read back of external gpio_ext_portX data timing.....	59
Fig 8-4 Interrupt RTL block diagram.....	60
Fig 8-5 Debounce RTL diagram	60
Fig 8-6 Debounce timing with asynchronous reset Flip-Flops	61
Fig 8-7 Synchronization and edge detect interrupt generation when GPIO_INT_BOTH_EDGE=0	62
Fig 8-8 Interrupt edge detection and interrupt clear timing when GPIO_SYNC_PA_INTERRUPTS = 1 (metastability included)	62
Fig 8-9 Interrupt edge detection and interrupt clear timing when GPIO_SYNC_PA_INTERRUPTS = 0 (metastability removed)	63
Fig 8-10 Write to interrupt clear register, coincident with detection of new interrupt	64
Fig 8-11 Synchronization and edge detect interrupt generation when GPIO_INT_BOTH_EDGE=0	64
Fig 8-12 Interrupt edge detection and interrupt clear timing when GPIO_SYNC_PA_INTERRUPTS = 1 and GPIO_INT_BOTH_EDGE=1 (metastability included)	65
Fig 8-13 Interrupt edge detection and interrupt clear timing when GPIO_SYNC_PA_INTERRUPTS = 0 and GPIO_INT_BOTH_EDGE=1 (metastability Removed).....	66
Fig 8-14 Level-sensitive interrupt RTL diagram.....	66
Fig 8-15 Active-low level-sensitive interrupt generation timing.....	67
Fig 8-16 Relationship between APB and APB slave data widths	67
Fig 9-1 Block diagram of DMAC	84
Fig 9-2 Peripheral-to-Peripheral DMA transfer on the same AHB layer	85
Fig 9-3 Peripheral-to-Memory DMA transfer on separate AHB layers	86
Fig 9-4 DMA Transfer Hierarchy for Non-Memory Peripherals	87
Fig 9-5 DMA transfer hierarchy for memory.....	87
Fig 9-6 Hardware handshaking interface	93
Fig 9-7 Burst transaction – pclk = hclk	94
Fig 9-8 Back-to-Back burst transactions – hclk = 2*per_clk.....	95
Fig 9-9 Single transaction	96
Fig 9-10 Burst followed by Back-to-Back single transactions.....	96
Fig 9-11 Early-Terminated burst transaction	96
Fig 9-12 Burst transaction ignored during active single transaction	97
Fig 9-13 Generation of dma_req and dma_single by source	97
Fig 9-14 Hardware handshaking interface	99
Fig 9-15 Burst transaction followed by single transaction that terminates block	99
Fig 9-16 Single transaction followed by burst transaction that terminates block	100
Fig 9-17 Breakdown of Block Transfer	102
Fig 9-18 Channel FIFO contents at times indicated in Fig 9-17	102
Fig 9-19 Breakdown of block transfer for DMAH_CH_FIFO_DEPTH = 8	103
Fig 9-20 Breakdown of block transfer where max_abrst = 2, Case 1.....	104
Fig 9-21 Channel FIFO contents at times indicated in Fig 9-20.....	104
Fig 9-22 Breakdown of block transfer where max_abrst = 2, Case 2.....	105
Fig 9-23 Channel FIFO contents at times indicated in Fig 9-22.....	105

Fig 9-24 Breakdown of block transfer	106
Fig 9-25 Source FIFO contents at time indicated in Fig 9-24.....	107
Fig 9-26 Source FIFO contents where watermark level is dynamically adjusted	107
Fig 9-27 Block transfer to destination	108
Fig 9-28 Block transfer up to time 't4'	110
Fig 9-29 Source, DMAC channel and destination FIFOs at time 't4' in Fig 9-26.....	110
Fig 9-30 FIFO status after early-terminated burst	111
Fig 9-31 Data loss when pre-fetching is enabled	113
Fig 9-32 Timing exception on dma_finish to source peripheral	114
Fig 9-33 Case of no data loss when pre-fetching is enabled.....	114
Fig 9-34 Source enters single transaction region when destination asserts dma_last[1]	115
Fig 9-35 Case where source does not enter single transaction region when destination asserts dma_last[1].....	116
Fig 9-36 Data loss when data pre-fetching is disabled.....	118
Fig 9-37 Transaction request through peripheral interrupt.....	119
Fig 9-38 Flow control configurations	120
Fig 9-39 Case 1 watermark levels where SSI.DMATDLR = 2	121
Fig 9-40 Case 2 watermark levels where SSI.DMATDLR = 6	122
Fig 9-41 SSI receive FIFO	123
Fig 9-42 Arbitration flow for master bus interface	126
Fig 9-43 Example of destination scatter transfer.....	128
Fig 9-44 Source gather when SGR.SGI = 0x1	129
Fig 9-45 Multi-block transfer using linked lists when DMAH_CHx_STAT_SRC set to true.....	172
Fig 9-46 Multi-block transfer using linked lists when DMAH_CHx_STAT_SRC set to false	172
Fig 9-47 Mapping of block descriptor (LLI) in memory to channel registers when DMAH_CHx_STAT_SRC set to true	172
Fig 9-48 Mapping of block descriptor (LLI) in memory to channel registers when DMAH_CHx_STAT_SRC set to false	173
Fig 9-49 Flowchart for DMA programming example	177
Fig 9-50 Multi-block with linked address for source and destination.....	180
Fig 9-51 Multi-block with linked address for source and destination where SARx and DARx between successive blocks are contiguous.....	181
Fig 9-52 DMA transfer flow for source and destination linked list address.....	182
Fig 9-53 Multi-block DMA transfer with source and destination address auto-reloaded	183
Fig 9-54 DMA transfer flow for source and destination address auto-reloaded	184
Fig 9-55 Multi-block DMA transfer with source address auto-reloaded and linked list destination address	186
Fig 9-56 DMA transfer flow for source address auto-reloaded and linked list destination address	187
Fig 9-57 Multi-block DMA transfer with source address auto-reloaded and contiguous destination address	189
Fig 9-58 DMA transfer flow for source address auto-reloaded and contiguous destination address	189
Fig 9-59 Multi-block DMA transfer with linked list source address and contiguous destination address.....	191
Fig 9-60 DMA transfer flow for source address auto-reloaded and contiguous destination address	192
Fig 10-1 Block diagram.....	194
Fig 10-2 TIM4 block diagram.....	196
Fig 10-3 Statistic pulse width mode diagram (positive edge of TRGI is active for capture).....	197
Fig 10-4 Statistic pulse number mode diagram (positive edge of TRGI is active for capture, ARR=E6).....	198
Fig 10-5 PWM timer block diagram	199
Fig 10-6 Counter timing diagram with prescaler division change from 1 to 2	200
Fig 10-7 Counter timing diagram with prescaler division change from 1 to 4	201
Fig 10-8 Counter timing diagram (internal clock divided by 1)	202
Fig 10-9 Counter timing diagram (internal clock divided by 2)	202
Fig 10-10 Counter timing diagram (internal clock divided by 4)	202
Fig 10-11 Counter timing diagram (internal clock divided by N)	203
Fig 10-12 Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded).....	203
Fig 10-13 Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)	204
Fig 10-14 Edge-aligned PWM waveforms (ARR=8, CCxP=0)	205
Fig 10-15 One-pulse mode timing.....	205

Fig 10-16 Block diagram.....	228
Fig 10-17 Synchronous data diagram.....	228
Fig 11-1 RTC block diagram.....	232
Fig 11-2 RTC prescale diagram.....	233
Fig 11-3 RTC clock select diagram.....	233
Fig 11-4 Calibration block diagram	234
Fig 11-5 xtal_req_32k block diagram	234
Fig 13-1 Block diagram of I ² C	247
Fig 13-2 Master/Slave and Transmitter/Receiver relationships	248
Fig 13-3 Data transfer on the I ² C bus	249
Fig 13-4 START and STOP conditions	250
Fig 13-5 7-bit address format.....	250
Fig 13-6 10-bit address format.....	250
Fig 13-7 Master-Transmitter protocol	251
Fig 13-8 Master-Receiver protocol	252
Fig 13-9 START BYTE transfer	252
Fig 13-10 General call address format	252
Fig 13-11 NULL DATA transfer format.....	253
Fig 13-12 IC_DATA_CMD register content.....	253
Fig 13-13 Master transmitter — Tx FIFO empties/STOP generation	253
Fig 13-14 Master receiver — Tx FIFO empties/STOP generation	254
Fig 13-15 Master transmitter — Restart bit of IC_DATA_CMD is set	254
Fig 13-16 Master receiver — Restart bit of IC_DATA_CMD is set.....	254
Fig 13-17 Master transmitter — Stop bit of IC_DATA_CMD set/Tx FIFO not empty	254
Fig 13-18 Master receiver — Stop bit of IC_DATA_CMD set/Tx FIFO not empty	255
Fig 13-19 Multiple master arbitration.....	255
Fig 13-20 Multi-Master clock synchronization.....	256
Fig 13-21 I ² C 8-bit FIFO content with transfer control register	261
Fig 14-1 UART block diagram	285
Fig 14-2 Clock structure of KM0 log UART Rx path	299
Fig 14-3 Clock structure of KM0 LUART Rx path	300
Fig 14-4 Clock structure of KM4 UART0 Rx path.....	300
Fig 14-5 Relationship between IrDA signal and UART signal	301
Fig 14-6 Signal connection in auto-flow control mode	301
Fig 14-7 DMA interface timing diagram.....	302
Fig 14-8 DMA interface timing diagram.....	302
Fig 15-1 IR signal model	303
Fig 15-2 IR Tx flow.....	303
Fig 15-3 IR Rx flow.....	304
Fig 15-4 IR block diagram.....	304
Fig 15-5 Tx output level.....	314
Fig 16-1 Key-Scan block diagram.....	315
Fig 16-2 Typical application setup with external keypad.....	316
Fig 16-3 Key-Scan flow	317
Fig 16-4 Key-Scan timing	318
Fig 16-5 Difference of FIFO items between two work modes	318
Fig 16-6 FIFO structure.....	319
Fig 16-7 Clock domain diagram.....	320
Fig 16-8 4*3 keypad example	321
Fig 16-9 Shadow key condition	321
Fig 16-10 Correct three-key condition	322
Fig 17-1 Audio codec diagram.....	330
Fig 17-2 Cap-less mode connection with headphone jack.....	334

Fig 17-3 Differential mode connection with headphone jack.....	335
Fig 17-4 Single-end mode connection with headphone jack.....	335
Fig 17-5 Line-in mode connection.....	336
Fig 17-6 Analog MIC single-end mode connection	336
Fig 17-7 Analog MIC differential mode connection	336
Fig 17-8 Digital MIC mono mode connection	337
Fig 17-9 Digital MIC stereo mode connection.....	337
Fig 17-10 Mono PDM format	337
Fig 17-11 Stereo PDM format	338
Fig 17-12 I ² S acting as PDM.....	338
Fig 18-1 Ameba-D ACC + AC architecture	382
Fig 18-2 ACC block diagram	383
Fig 18-3 I ² S audio data format	386
Fig 18-4 Left-Justified data format.....	387
Fig 18-5 PCM mode B data format.....	387
Fig 18-6 PCM mode B-N data format.....	387
Fig 18-7 PCM mode A data format	387
Fig 18-8 PCM mode A-N data format.....	388
Fig 18-9 SI write timing	388
Fig 18-10 SI read timing	389
Fig 18-11 ACC clock architecture	389
Fig 19-1 SPI block diagram	396
Fig 19-2 SPI Serial Format (SCPH = 0).....	398
Fig 19-3 SPI Serial Format Continuous Transfers (SCPH = 0 and SS toggling)	398
Fig 19-4 SPI Serial Format Continuous Transfers (SCPH = 0 and SS not-toggling)	398
Fig 19-5 SPI Serial Format (SCPH = 1).....	399
Fig 19-6 SPI Serial Format Continuous Transfers (SCPH = 1).....	399
Fig 19-7 Maximum sclk_out/ssi_clk Ratio.....	400
Fig 19-8 SPI Configured as master device	402
Fig 19-9 Effects of round trip routing delays on sclk_out signal	402
Fig 20-1 MCU I/F + LCM with GRAM	422
Fig 20-2 RGB I/F + LCM without GRAM.....	423
Fig 20-3 LCD block diagram.....	423
Fig 20-4 Two data paths.....	424
Fig 20-5 MCU I/O mode application scenario	424
Fig 20-6 DMA mode application scenario	425
Fig 20-7 MCU interface	425
Fig 20-8 MCU I/F command setting timing parameters	426
Fig 20-9 MCU I/F data writing timing parameters	426
Fig 20-10 MCU I/F read command timing parameters	426
Fig 20-11 MCU VSYNC mode timing	427
Fig 20-12 MCU TE mode timing	427
Fig 20-13 MCU TE mode frame synchronization	427
Fig 20-14 8080 I/F 8-bit output.....	428
Fig 20-15 8080 I/F 16-bit output.....	428
Fig 20-16 RGB interface	429
Fig 20-17 RGB timing.....	429
Fig 20-18 RGB DE mode timing	430
Fig 20-19 RGB I/F 6-bit output	430
Fig 20-20 RGB I/F 16-bit output	431
Fig 20-21 LED interface	432
Fig 20-22 LED control timing.....	433
Fig 20-23 LED control diagram.....	434

Fig 20-24 LED color mapping: single color and single channel	434
Fig 20-25 LED color capping: single color and two channels	435
Fig 20-26 LED color mapping: two colors and single channel	435
Fig 20-27 LED color mapping: two colors and two channels	436
Fig 20-28 LED color mapping: three colors and single channel	436
Fig 20-29 LED color mapping: three colors and two channels	437
Fig 20-30 Max. supported resolutions for LCDC	438
Fig 21-1 Q-Decoder application scenario	454
Fig 21-2 Quadrature signal	455
Fig 21-3 Q-Decoder block diagram	455
Fig 21-4 Q-Decoder phase state	456
Fig 21-5 Position count state when CNT_SC = 0	456
Fig 21-6 Position count state when CNT_SC = 1	457
Fig 21-7 Position counter reset on index pulse (forward direction)	457
Fig 21-8 Position counter reset on index pulse (reverse direction)	457
Fig 21-9 IDX_INV = 0, position counter is reset on (PHA, PHB) = (1, 0)	458
Fig 21-10 IDX_INV = 1, position counter is reset on (PHA, PHB) = (1, 0)	458
Fig 21-11 Auto-index mechanism when IDX_INV = 0	459
Fig 21-12 Auto-index mechanism when IDX_INV = 1	459
Fig 21-13 Rotation count when RC_MOD = 1	460
Fig 21-14 Velocity measurement unit timing flow	461
Fig 22-1 I ² S mono/stereo audio-out interface configuration	473
Fig 22-2 I ² S 5.1 channel audio-out interface configuration	474
Fig 22-3 Signal lines in I ² S data format	474
Fig 22-4 Transmitter as the master	475
Fig 22-5 Receiver as the master	475
Fig 22-6 Controller as the master	476
Fig 22-7 I ² S Philips standard	476
Fig 22-8 Left-justified standard	476
Fig 22-9 Right-justified standard	477
Fig 22-10 I ² S clock tree	477
Fig 22-11 Memory block	478
Fig 22-12 FIFO allocation of mono channel (sample bit = 16-bit)	479
Fig 22-13 FIFO allocation of mono channel (sample bit = 32-bit)	479
Fig 22-14 FIFO allocation of stereo channel (sample bit = 16-bit)	480
Fig 22-15 FIFO allocation of stereo channel (sample bit = 24-bit)	480
Fig 22-16 FIFO allocation of stereo channel (sample bit = 32-bit)	480
Fig 22-17 FIFO allocation of 5.1 channel (sample bit = 16-bit)	481
Fig 22-18 FIFO allocation of 5.1 channel (sample bit = 24-bit)	481
Fig 22-19 FIFO allocation of 5.1 channel (sample bit = 32-bit)	481

Conventions

The following abbreviations are used to illustrate the access for registers, and they represent the same meaning in different registers.

R/W	Read/write. Software can read and write to this bit.
R	Read-only. Software can only read this bit.
W	Write-only. Software can only write to this bit, reading this bit returns the reset value.
R/W/EC	EC: external clear. Software can read and write to this bit, hardware can also clear this bit.
R/W/ES	ES: external set. Software can read and write to this bit, hardware can also write to this bit.
R/W1C	Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value.
R/W1only	Software can read and write 1 to this bit only. Writing 0 has no effect on the bit value.
R/W1S	
R/W (SHW)	
R/W/AC	
W1C	Software can clear this bit by writing 1.
WA0	Hardware will automatically clear to 0 when software has written to this bit.
N/A	Used for the reserved bit which should not be concerned by users.

1 Product Overview

1.1 General Description

Ameba-D (RTL872xD) is a highly integrated single-chip low power dual bands (2.4GHz and 5GHz) Wireless LAN (WLAN) and Bluetooth Low Energy (BLE5.0) communication controller. It consists of a high performance MCU (Armv8-M, Cortex-M33 instruction set compatible) called Real-M300 (or KM4 thereafter) and a low power MCU (Armv8-M, Cortex-M23 instruction set compatible) called Real-M200 (or KM0 thereafter), WLAN (802.11 a/b/g/n) MAC, an 1T1R capable WLAN baseband, RF, Bluetooth and peripherals.

High speed connectivity interfaces – SDIO and USB are provided. There are also audio codec, Key-Scan and touch keys integrated into this IC. Besides, flexible design can configure GPIO to different functions according to different applications.

Ameba-D also integrates memories (ROM/SRAM/PSRAM) for IoT (Internet of Things) Wi-Fi protocol functions and applications. The user-friendly development kits – SDK and HDK are provided to customers for developing IoT applications.

The KM4 MCU is a 32-bit core offering system enhancements, such as low power consumption, enhanced debug features, floating point computation, DSP instructions and a high level of support block integration. The KM4 MCU incorporates a 3-stage pipeline.

The KM0 coprocessor is an energy-efficient and easy-to-use 32-bit core which is code-compatible and tool-compatible with the KM4 core. The KM0 coprocessor offers up to 20MHz performance with a simple instruction set and a reduced code size.

1.2 System Architecture

The system architecture of Ameba-D is shown in Fig 1-1.

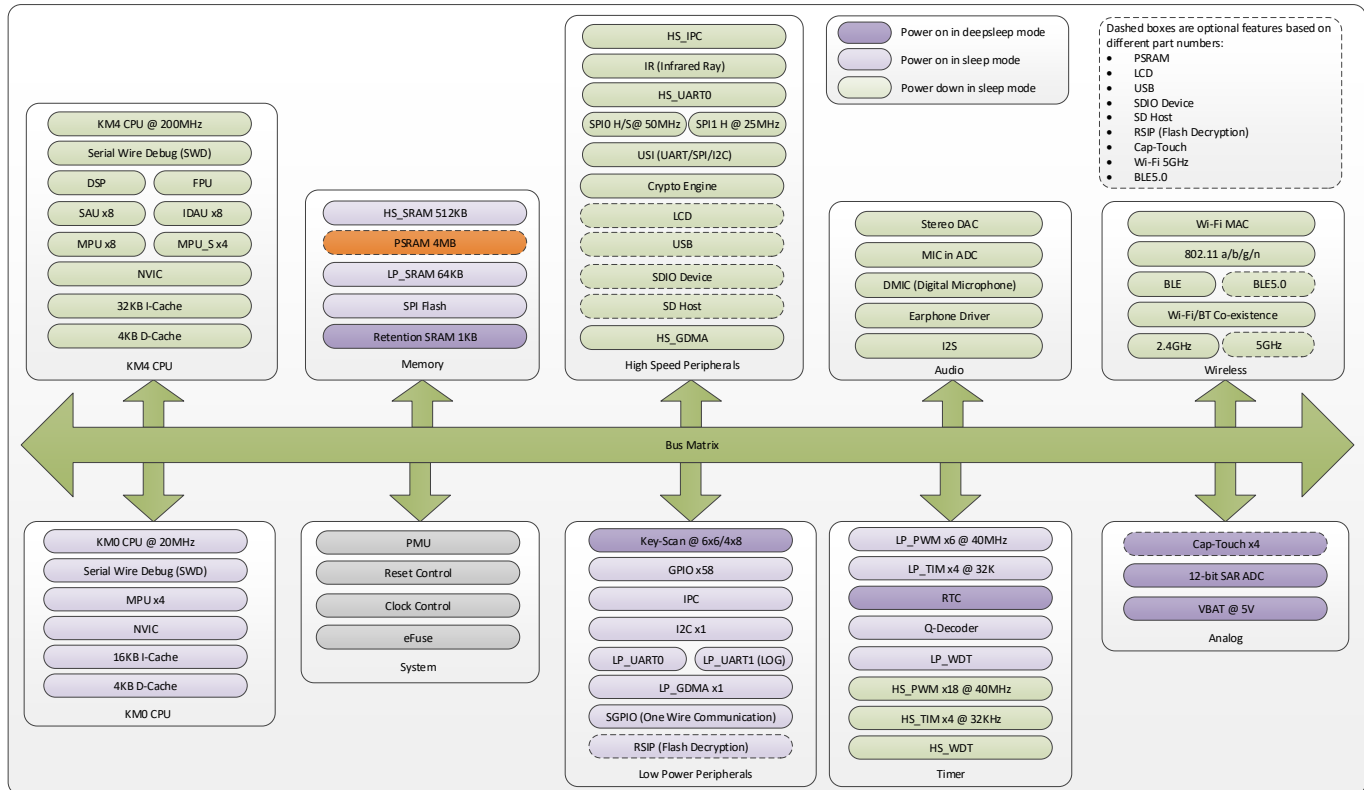


Fig 1-1 System architecture

In Ameba-D, the main system consists of 32-bit multilayer AXI bus matrix that interconnects all masters and slaves. The bus matrix provides access from a master to a slave, enables concurrent access and efficient operation even when several high-speed peripherals work simultaneously.

A multilayer AXI bus matrix connects the CPU buses and other bus masters to peripherals in a flexible manner, which can optimize performance because it allows peripherals on different slave ports of the bus matrix to be accessed simultaneously by different bus masters.

APB peripherals are connected to the AXI bus matrix via APB buses using separated slave ports from the multilayer AXI bus matrix. This allows for better performance by reducing collisions between the CPU and the DMA controller, and also for peripherals on the asynchronous bridge to have a fixed clock that doesn't track the system clock.

2 Memory Organization

2.1 Introduction

Ameba-D incorporates several distinct memory regions. Program memory, data memory, registers, and I/O ports are organized within the same linear 4Gbytes address space. The bytes are coded in memory in Little-Endian format.

The addressable memory space is divided into multiple main blocks, as shown in Table 2-1. All the memory areas that are not allocated to on-chip memories and peripherals are considered “RSVD” (reserved).

Table 2-1 Address space main blocks

Base Address	Top Address	Size	Function	Description
0x0000_0000	0x0001_FFFF	128KB	KM0 ITCM ROM (actually 96KB)	32MB: KM0 Memory Address
0x0002_0000	0x0002_7FFF	32KB	KM0 DTCM ROM (actually 16KB)	
0x0002_8000	0x0007_FFFF	352KB	RSVD	
0x0008_0000	0x0008_FFFF	64KB	KM0 SRAM	
0x0009_0000	0x000B_FFFF	192KB	RSVD	
0x000C_0000	0x000C_3FFF	16KB	Retention SRAM (actually 1KB) (the same port with KM0 SRAM)	
0x000C_4000	0x000F_FFFF	240KB	RSVD	
0x0010_0000	0x01FF_FFFF	31MB	RSVD	
0x0200_0000	0x07FF_FFFF	96MB	External PSRAM	224MB: External Memory Address
0x0800_0000	0x0FFF_FFFF	128MB	External FLASH	
0x1000_0000	0x1007_FFFF	512KB	KM4 SRAM	256MB: KM4 Memory Address
0x1008_0000	0x100D_FFFF	384KB	RSVD	
0x100E_0000	0x100F_FFFF	128KB	Extension SRAM (the same port with KM4 SRAM)	
0x1010_0000	0x101B_FFFF	768KB	KM4 ITCM ROM (actually 256KB)	
0x101C_0000	0x101D_7FFF	96KB	KM4 DTCM ROM	
0x101D_8000	0x101F_FFFF	160KB	RSVD	
0x1020_0000	0x1FFF_FFFF	254MB	RSVD	
0x2000_0000	0x3FFF_FFFF	512MB	RSVD	Reserved
0x4000_0000	0x47FF_FFFF	128MB	KM4 Peripherals	128MB: KM4 Peripherals Address
0x4800_0000	0x4FFF_FFFF	128MB	KM0 Peripherals	128MB: KM0 Peripherals Address
0x5000_0000	0x57FF_FFFF	128MB	KM4 Peripherals Secure	128MB: KM4 Peripherals Secure Address
0x5800_0000	0xFFFF_FFFF	2688MB	RSVD	Reserved

For the detailed mapping of available memory and register areas, refer to the following sections.

2.2 KM4 Memory

2.2.1 Memory Map and Register Boundary Addresses

Table 2-2 gives the boundary addresses of the peripherals available in the KM4 devices.

Table 2-2 KM4 register boundary addresses

Port Name	Security	Base Address	Top Address	Size
KM4_SRAM1	IDAU	0x1000_0000	0x1003_FFFF	256KB
KM4_SRAM2	IDAU	0x1004_0000	0x1007_FFFF	256KB
Extension SRAM	IDAU	0x100E_0000	0x100F_FFFF	128KB
PSRAM Memory	IDAU	0x0200_0000	0x07FF_FFFF	96MB
HS_SYSON	Non-Secure	0x4000_0000	0x4000_0FFF	4KB
	Secure	0x5000_0000	0x5000_0FFF	4KB

HS_TIM0 ~ 3/4/5	Non-Secure	0x4000_2000	0x4000_2FFF	4KB
HS_UART0	Non-Secure	0x4000_4000	0x4000_4FFF	4KB
HS_IPC	Non-Secure	0x4000_6000	0x4000_6FFF	4KB
HS_USI	Non-Secure	0x4000_8000	0x4000_83FF	1KB
HS_UART1 (Bluetooth)	Non-Secure	0x4000_A000	0x4000_AFFF	4KB
RXI300_KM4	Non-Secure	0x4000_C000	0x4000_CFFF	4KB
	Secure	0x5000_C000	0x5000_CFFF	4KB
HS_SPI1	Non-Secure	0x4000_E000	0x4000_E7FF	2KB
Audio Codec	Non-Secure	0x4001_0000	0x4001_0FFF	4KB
HS_IR	Non-Secure	0x4001_2000	0x4001_2FFF	4KB
PSRAM Controller	Non-Secure	0x4001_4000	0x4001_4FFF	4KB
I ² S	Non-Secure	0x4002_0000	0x4002_03FF	1KB
Secure Engine	Non-Secure	0x4002_2000	0x4002_5FFF	16KB
	Secure	0x5002_2000	0x5002_5FFF	16KB
SDIO Host	Non-Secure	0x4002_6000	0x4002_9FFF	16KB
HS_GDMA0	Non-Secure	0x4002_A000	0x4002_BFFF	8KB
	Secure	0x5002_A000	0x5002_BFFF	8KB
SDIO Device	Non-Secure	0x4002_C000	0x4002_FFFF	16KB
USB	Non-Secure	0x4004_0000	0x4006_FFFF	192KB
LCD Controller	Non-Secure	0x4007_0000	0x4007_4FFF	20KB
HS_SPI0	Non-Secure	0x4007_8000	0x4007_87FF	2KB
Wi-Fi	Non-Secure	0x4008_0000	0x400A_FFFF	192KB
KM0 BRG	Non-Secure	0x0008_0000	0x0008_FFFF	64KB
		0x000C_0000	0x000C_3FFF	16KB
		0x4800_0000	0x4803_FFFF	256KB
Flash Controller	Non-Secure	0x4808_0000	0x4808_0FFF	4KB
Flash Memory	IDAU	0x0800_0000	0x0FFF_FFFF	128MB

2.2.2 Embedded SRAM

The KM4 contains up to a total 512KB of contiguous, on-chip static RAM memory. This embedded SRAM can be accessed as bytes (8 bits), half-words (16 bits) or full words (32 bits). It is divided into the following two blocks which can be accessed by both KM4 and KM0.

- KM4 SRAM1 (up to 256KB)
- KM4 SRAM2 (up to 256KB)

Dividing SRAM into two slave ports allows user's program to potentially obtain better performance. For example, simultaneous access to SRAM1 by the CPU and by the system DMA controller does not result in any bus stalls for either master.

Generally speaking, the CPU reads or writes all peripheral data at some point, even when all such data is read from or sent to a peripheral by DMA. So, minimizing stalls is likely to involve putting data to/from different peripherals in RAM on each port.

Alternatively, sequences of data from the same peripheral can be alternated between RAM on each port. This could be helpful if DMA fills or empties a RAM buffer, and signals the CPU before proceeding on to a second buffer. The CPU then tends to access the data while the DMA is using the other RAM.

In power domains, the entire SRAM is also divided into three blocks:

- SRAM_PD1 (up to 256KB)
- SRAM_PD2 (up to 128KB)
- SRAM_PD3 (up to 128KB)

Each block can be disabled or enabled individually in the Power Management Unit (PMU) block to save power, and the entire SRAM can also keep power for quickly resuming from sleep mode when system enters sleep mode.

2.2.3 Extension SRAM

When Bluetooth is disabled, more 64KB SRAM will be extended. This SRAM can also be accessed by both KM4 and KM0, up to 50MHz*32 bits.

2.3 KM0 Memory

2.3.1 Memory Map and Register Boundary Addresses

Table 2-3 gives the boundary addresses of the peripherals available in the KM0 devices.

Table 2-3 KM0 register boundary addresses

Port Name	Base Address	Top Address	Size
KM0_SRAM	0x0008_0000	0x0008_FFFF	64KB
1KB Retention SRAM	0x000C_0000	0x000C_3FFF	16KB
KM4 BRG	0x1000_0000	0x1007_FFFF	512KB
	0x4000_0000	0x4007_FFFF	512KB
Wi-Fi FW	0x4008_0000	0x400A_FFFF	192KB
LP_SYSON	0x4800_0000	0x4800_0FFF	4KB
LP_TIM0 ~ 3/4/5	0x4800_2000	0x4800_2FFF	4KB
LP_RTC	0x4800_4000	0x4800_43FF	1KB
LP_IPC	0x4800_6000	0x4800_63FF	1KB
Key-Scan	0x4800_A000	0x4800_A3FF	1KB
I ² C0	0x4800_C000	0x4800_C3FF	1KB
UART3	0x4800_E000	0x4800_E3FF	1KB
LP_GDMA0	0x4801_0000	0x4801_07FF	2KB
UART2 (LOGUART)	0x4801_2000	0x4801_23FF	1KB
GPIOA/B	0x4801_4000	0x4801_47FF	2KB
RXI300_KM0	0x4801_8000	0x4801_8FFF	4KB
SGPIO	0x4801_A000	0x4801_AFFF	4KB
Cap-Touch	0x4801_C000	0x4801_C7FF	2KB
ADC	0x4801_C800	0x4801_CBFF	1KB
Comparator	0x4801_CC00	0x4801_CFFF	1KB
Q-Decoder	0x4801_E000	0x4801_EFFF	4KB
Flash Controller	0x4808_0000	0x4808_0FFF	4KB
Flash Memory	0x0800_0000	0x0FFF_FFFF	128MB

2.3.2 Embedded SRAM

The KM0 features 64KB of system SRAM. The embedded SRAM can be accessed as bytes (8 bits), half-words (16 bits) or full words (32 bits).

This SRAM can be accessed by both KM4 and KM0.

2.4 Retention SRAM

Ameba-D features 1KB of retention SRAM in order to allow saving data with minimal power usage during deepsleep mode.

This SRAM can be accessed by both KM4 and KM0.

2.5 SPI Flash Memory

The SPI Flash Controller (SPIC) manages CPU I-Code and D-Code accesses to the Flash memory. It implements the erase and program Flash memory operations, and the read/write protection mechanisms. It accelerates code execution with a system of instruction prefetch and cache lines.

2.6 PSRAM

4MB 8IO DDR PSRAM is included in Ameba-D, up to 50MHz DDR.

3 Memory Protection Unit (MPU)

The KM4 and KM0 processor both have a memory protection unit (MPU) that provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis. Such requirements are critical in many embedded applications.

The MPU needs to be programmed and enabled before use. If the MPU is not enabled, the memory system behavior is the same as that no MPU is present.

The MPU can improve the reliability of an embedded system by:

- Preventing user applications from corrupting data used by the operating system
- Separating data between processing tasks by blocking tasks from accessing others' data
- Allowing memory regions to be defined as read-only so that vital data can be protected
- Detecting unexpected memory accesses (for example, stack corruption)

In addition, the MPU can also be used to define memory access characteristics such as caching and buffering behaviors for different regions. The MPU sets up the protection by defining the memory map as a number of regions. As Table 3-1 shows, up to eight regions can be defined, but it is also possible to define a default background memory.

Table 3-1 MPU entries

CPU	Secure	Region Number
KM4	Non-Secure MPU	x8
	Secure MPU	x4
KM0	Non-Secure MPU	x4

3.1 Register Map

Table 3-2 provides the details of the MPU register map. All registers in MPU are addressed at 32-bit boundary. Where the physical size of any register is less than 32-bit wide, the upper unused bits of the 32-bit boundary are reserved. Writing to these bits has no effect; reading from these bits returns 0.

Table 3-2 Register map of MPU

Name	Address Offset	Description
MPU_TYPE	0xE000_ED90	MPU Type Register
MPU_CTRL	0xE000_ED94	MPU Control Register
MPU_RNR	0xE000_ED98	MPU Region Number Register
MPU_RBAR	0xE000_ED9C	MPU Region Base Address Register
MPU_RLAR	0xE000_EDA0	MPU Region Limit Address Register
MPU_RBAR_A<n>	0xE000_EDA4	MPU Region Base Address Register Alias ($n = \{1, 2, 3\}$)
MPU_RLAR_A<n>	0xE000_EDA8	MPU Region Limit Address Register Alias ($n = \{1, 2, 3\}$)
MPU_MAIR0	0xE000_EDC0	MPU Memory Attribute Indirection Register 0
MPU_MAIR1	0xE000_EDC4	MPU Memory Attribute Indirection Register 1

3.2 Register Field Description

3.2.1 MPU_TYPE

The MPU_TYPE characteristics are:

- **Purpose:** The MPU Type Register indicates how many regions the MPU supports for the selected security state.
- **Usage constraints:**
 - Privileged access is permitted only. Unprivileged access generates a BusFault.
 - This register is word accessible only. Half-word and byte accesses are unpredictable.
- **Configurations:** This register is always implemented.

- **Attributes:**
 - 32-bit read-only register located at 0xE000_ED90.
 - Secure software can access the Non-Secure view of this register via MPU_TYPE_NS located at 0xE002_ED90. The location 0xE002_ED90 is reserved to software executing in Non-Secure state and the debugger.
 - This register is banked between security states.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DERGION								RSVD							SEPARATE
R															R

Bit	Name	Access	Description
31:16	RSVD	N/A	Reserved
15:8	DREGION	R	Data regions. Number of regions supported by the MPU. This field reads as an implementation defined value. If this field reads as 0, the PE doesn't implement a MPU for the selected security state.
7:1	RSVD	N/A	Reserved
0	SEPARATE	R	Separate. Indicates support for separated instructions and data address regions. ARMv8-M only supports unified MPU regions. This bit reads as 0.

3.2.2 MPU_CTRL

The MPU_CTRL register characteristics are:

- **Purpose:** Enables the MPU, and when the MPU is enabled, controls whether the default memory map is enabled as a background region for privileged access, and whether the MPU is enabled for HardFaults, NMIs, and exception handlers when FAULTMASK is set to 1.
- **Usage constraints:**
 - Privileged access is permitted only. Unprivileged access generates a BusFault.
 - This register is word accessible only. Half-word and byte accesses are unpredictable.
- **Configurations:** This register is always implemented.
- **Attributes:**
 - 32-bit read/write register located at 0xE000_ED94.
 - Secure software can access the Non-Secure view of this register via MPU_CTRL_NS located at 0xE002_ED94. The location 0xE002_ED94 is RES0 to software executing in Non-Secure state and the debugger.
 - This register is banked between security states.

31	30	29	...	5	4	3	2	1	0
RSVD							PRIVDEFENA	HFNMENA	ENABLE
							R/W	R/W	R/W

Bit	Name	Access	Description
31:3	RSVD	N/A	Reserved
2	PRIVDEFENA	R/W	Privileged default enable. Controls whether the default memory map is enabled for privileged software. <ul style="list-style-type: none"> ● 0: Use of default memory map disabled. ● 1: Use of default memory map enabled for privileged code. Note: <ul style="list-style-type: none"> ● When the ENABLE bit is set to 0, the PE ignores this bit. ● If no regions are enabled and the PRIVDEFENA and ENABLE bits are set to 1, only privileged code can execute from the system address map. ● If no MPU regions are implemented, this bit is reserved. ● This bit resets to 0 on a Warm reset.
1	HFNMENA	R/W	HardFault, NMI enable. Controls whether handlers executing with priority less than 0 access memory with the MPU enabled or disabled. This applies to HardFaults, NMIs, and exception handlers when FAULTMASK is set to 1.

			<ul style="list-style-type: none"> 0: MPU is disabled for these handlers. 1: MPU is enabled for these handlers. Note: <ul style="list-style-type: none"> If this bit is set to 1 when ENABLE is set to 0, behavior is unpredictable. If no MPU regions are implemented, this bit is reserved. This bit resets to 0 on a Warm reset.
0	ENABLE	R/W	Enable. Enables the MPU. <ul style="list-style-type: none"> 0: The MPU is disabled. 1: The MPU is enabled. Disabling the MPU means that privileged and unprivileged accesses use the default memory map. Note: <ul style="list-style-type: none"> If no MPU regions are implemented, this bit is reserved. This bit resets to 0 on a Warm reset.

3.2.3 MPU_RNR

The MPU_RNR characteristics are:

- **Purpose:** Selects the region currently accessed by MPU_RBAR and MPU_RLAR.
- **Usage constraints:**
 - Privileged access is permitted only. Unprivileged access generates a BusFault.
 - This register is word accessible only. Half-word and byte accesses are unpredictable.
- **Configurations:** This register is always implemented.
- **Attributes:**
 - 32-bit read/write register located at 0xE000_ED98.
 - Secure software can access the Non-Secure view of this register via MPU_RNR_NS located at 0xE002_ED98. The location 0xE002_ED98 is reserved to software executing in Non-Secure state and the debugger.
 - This register is banked between security states.

31	30	29	...	10	9	8	7	6	5	...	2	1	0
RSVD							REGION						
							R/W						

Bit	Name	Access	Description
31:8	RSVD	N/A	Reserved
7:0	REGION	R/W	Region number. Indicates the memory region accessed by MPU_RBAR and MPU_RLAR. Note: <ul style="list-style-type: none"> If no MPU regions are implemented, this field is reserved. Writing a value corresponding to an unimplemented region is constrained unpredictable. This field resets to an unknown value on a Warm reset.

3.2.4 MPU_RBAR

The MPU_RBAR register characteristics are:

- **Purpose:** Provides indirect read and write access to the base address of the currently selected MPU region for the selected security state.
- **Usage constraints:**
 - Privileged access is permitted only. Unprivileged access generates a BusFault.
 - This register is word accessible only. Half-word and byte accesses are unpredictable.
- **Configurations:** This register is always implemented.
- **Attributes:**
 - 32-bit read/write register located at 0xE000_ED9C.
 - Secure software can access the Non-Secure view of this register via MPU_RBAR_NS located at 0xE002_ED9C. The location 0xE002_ED9C is reserved to software executing in Non-Secure state and the debugger.
 - This register is banked between security states.
- **Preface:** This register provides access to the configuration of the MPU region selected by MPU_RNR.REGION for the appropriate security state. The field description applies to the currently selected region.

31	30	29	...	7	6	5	4	3	2	1	0
BASE							SH		AP[2:1]		XN
R/W							R/W		R/W		R/W

Bit	Name	Access	Description
31:5	BASE	R/W	Base address. Contains bit[31:5] of the lower inclusive limit of the selected MPU memory region. This value is zero-extended to provide the base address to be checked against. This field resets to an unknown value on a Warm reset.
4:3	SH	R/W	Shareability. Defines the shareable domain of this region for normal memory. <ul style="list-style-type: none"> 00: Non-shareable 10: Outer Shareable 11: Inner Shareable Others: Reserved For any type of device memory, the value of this field is ignored. This field resets to an unknown value on a Warm reset.
2:1	AP[2:1]	R/W	Access permissions. Defines the access permissions for this region. <ul style="list-style-type: none"> 00: Read/write by privileged code only. 01: Read/write by any privileged level. 10: Read-only by privileged code only. 11: Read-only by any privileged level. This field resets to an unknown value on a Warm reset.
0	XN	R/W	Execute never. Defines whether the code can be executed from this region. <ul style="list-style-type: none"> 0: Execution is only permitted if read permitted. 1: Execution is not permitted. This bit resets to an unknown value on a Warm reset.

3.2.5 MPU_RLAR

The MPU_RLAR characteristics are:

- **Purpose:** Provides indirect read and write access to the limit address of the currently selected MPU region for the selected security state.
- **Usage constraints:**
 - Privileged access is permitted only. Unprivileged access generates a BusFault.
 - This register is word accessible only. Half-word and byte accesses are unpredictable.
- **Configurations:** This register is always implemented.
- **Attributes:**
 - 32-bit read/write register located at 0xE000_EDA0.
 - Secure software can access the Non-Secure view of this register via MPU_RLAR_NS located at 0xE002_EDA0. The location 0xE002_EDA0 is reserved to software executing in Non-Secure state and the debugger.
 - This register is banked between Security states.
- **Preface:** This register provides access to the configuration of the MPU region selected by MPU_RNR.REGION for the appropriate security state. The field description applies to the currently selected region.

31	30	29	...	7	6	5	4	3	2	1	0
LIMIT							RSVD	AttrIndx		EN	
R/W								R/W		R/W	

Bit	Name	Access	Description
31:5	LIMIT	R/W	Limit address. Contains bit[31:5] of the upper inclusive limit of the selected MPU memory region. This value is post-fixed with 0x1F to provide the limit address to be checked against. This field resets to an unknown value on a Warm reset.
4	RSVD	N/A	Reserved
3:1	AttrIndx	R/W	Attribute index. Associates a set of attributes in the MPU_MAIR0 and MPU_MAIR1 register. This field resets to an unknown value on a Warm reset.
0	EN	R/W	Enable. Region enable. <ul style="list-style-type: none"> 0: Region is disabled. 1: Region is enabled. This bit resets to 0 on a Warm reset.

3.2.6 MPU_RBAR_A<n>

The MPU_RBAR_A<n> ($n = \{1, 2, 3\}$) characteristics are:

- **Purpose:** Provides indirect read and write access to the base address of the MPU region selected by MPU_RNR[7:2]:(n[1:0]) for the selected security state.
- **Usage constraints:**
 - Privileged access is permitted only. Unprivileged access generates a BusFault.
 - This register is word accessible only. Half-word and byte accesses are unpredictable.
- **Configurations:**
 - Present only if the Main Extension is implemented.
 - Reserved if the Main Extension is not implemented.
- **Attributes:**
 - 32-bit read/write register located at 0xE000_EDA4 + 8(n-1).
 - Secure software can access the Non-Secure view of this register via MPU_RBAR_A<n>_NS located at 0xE002_EDA4 + 8(n-1). The location 0xE002_EDA4 + 8(n-1) is reserved to software executing in Non-Secure state and the debugger.
 - This register is banked between Security states.
- **Preface:** This register is an alias of the MPU_RBAR register and provides access to the configuration of the MPU region selected by MPU_RNR.REGION, while REGION[1:0] has been set to n[1:0].

31	30	29	...	7	6	5	4	3	2	1	0
BASE							SH		AP[2:1]		XN
R/W							R/W		R/W		R/W

Bit	Name	Access	Description
31:5	BASE	R/W	Base address. Contains bit[31:5] of the lower inclusive limit of the selected MPU memory region. This value is zero-extended to provide the base address to be checked against. This field resets to an unknown value on a Warm reset.
4:3	SH	R/W	Shareability. Defines the shareable domain of this region for normal memory. <ul style="list-style-type: none"> ● 00: Non-shareable ● 10: Outer Shareable ● 11: Inner Shareable ● Others: Reserved For any type of device memory, the value of this field is ignored. This field resets to an unknown value on a Warm reset.
2:1	AP[2:1]	R/W	Access permissions. Defines the access permissions for this region. <ul style="list-style-type: none"> ● 00: Read/write by privileged code only. ● 01: Read/write by any privileged level. ● 10: Read-only by privileged code only. ● 11: Read-only by any privileged level. This field resets to an unknown value on a Warm reset.
0	XN	R/W	Execute never. Defines whether the code can be executed from this region. The possible values of this bit are: <ul style="list-style-type: none"> ● 0: Execution is only permitted if read permitted. ● 1: Execution is not permitted. This bit resets to an unknown value on a Warm reset.

3.2.7 MPU_RLAR_A<n>

The MPU_RLAR_A<n> ($n = \{1, 2, 3\}$) characteristics are:

- **Purpose:** Provides indirect read and write access to the limit address of the currently selected MPU region selected by MPU_RNR[7:2]:(n[1:0]) for the selected security state.
- **Usage constraints:**
 - Privileged access is permitted only. Unprivileged access generates a BusFault.
 - This register is word accessible only. Half-word and byte accesses are unpredictable.
- **Configurations:**
 - Present only if the Main Extension is implemented.
 - Reserved if the Main Extension is not implemented.

- **Attributes:**
 - 32-bit read/write register located at $0xE000_EDA8 + 8(n-1)$.
 - Secure software can access the Non-Secure view of this register via $MPU_RLAR_A<n>_NS$ located at $0xE002_EDA8 + 8(n-1)$. The location $0xE002_EDA8 + 8(n-1)$ is reserved to software executing in Non-Secure state and the debugger.
 - This register is banked between Security states.
- **Preface:** This register is an alias of the MPU_RLAR register and provides access to the configuration of the MPU region selected by MPU_RNR.REGION, while REGION[1:0] has been set to n[1:0].

31	30	29	...	7	6	5	4	3	2	1	0
LIMIT							RSVD	AttrIndx		EN	
R/W								R/W		R/W	

Bit	Name	Access	Description
31:5	LIMIT	R/W	Limit address. Contains bit[31:5] of the upper inclusive limit of the selected MPU memory region. This value is post-fixed with 0x1F to provide the limit address to be checked against. This field resets to an unknown value on a Warm reset.
4	RSVD	N/A	Reserved
3:1	AttrIndx	R/W	Attribute index. Associates a set of attributes in the MPU_MAIRO and MPU_MAIR1 register. This field resets to an unknown value on a Warm reset.
0	EN	R/W	Enable. Region enable. <ul style="list-style-type: none"> ● 0: Region is disabled. ● 1: Region is enabled. This bit resets to 0 on a Warm reset.

3.2.8 MPU_MAIRO

The MPU_MAIRO characteristics are:

- **Purpose:** Along with MPU_MAIR1, provides the memory attribute encoding corresponding to the AttrIndx values.
- **Usage constraints:**
 - Privileged access is permitted only. Unprivileged access generates a BusFault.
 - This register is word accessible only. Half-word and byte accesses are unpredictable.
- **Configurations:** This register is always implemented.
- **Attributes:**
 - 32-bit read/write register located at $0xE000_EDC0$.
 - Secure software can access the Non-Secure view of this register via MPU_MAIRO_NS located at $0xE002_EDC0$. The location $0xE002_EDC0$ is reserved to software executing in Non-Secure state and the debugger.
 - This register is banked between Security states.
- **Preface:** This register is reserved if no MPU regions are implemented in the corresponding security state.

31	30	29	28	27	26	25	24
ATTR3							
R/W							
23	22	21	20	19	18	17	16
ATTR2							
R/W							
15	14	13	12	11	10	9	8
ATTR1							
R/W							
7	6	5	4	3	2	1	0
ATTR0							
R/W							

Bit	Name	Access	Description
31:24	ATTR3	R/W	Memory attributes encoding for MPU regions with an AttrIndx of 3. For the possible values of this field, refer to MAIR_ATTR for encoding. This field resets to an unknown value on a Warm reset.
23:16	ATTR2	R/W	Memory attributes encoding for MPU regions with an AttrIndx of 2.

			For the possible values of this field, refer to MAIR_ATTR for encoding. This field resets to an unknown value on a Warm reset.
15:8	ATTR3	R/W	Memory attributes encoding for MPU regions with an AttrIdx of 1. For the possible values of this field, refer to MAIR_ATTR for encoding. This field resets to an unknown value on a Warm reset.
7:0	ATTR0	R/W	Memory attributes encoding for MPU regions with an AttrIdx of 0. For the possible values of this field, refer to MAIR_ATTR for encoding. This field resets to an unknown value on a Warm reset.

3.2.9 MPU_MAIR1

The MPU_MAIR1 characteristics are:

- **Purpose:** Along with MPU_MAIRO, provides the memory attribute encoding corresponding to the AttrIdx values.
- **Usage constraints:**
 - Privileged access is permitted only. Unprivileged access generates a BusFault.
 - This register is word accessible only. Half-word and byte accesses are unpredictable.
- **Configurations:** This register is always implemented.
- **Attributes:**
 - 32-bit read/write register located at 0xE000_EDC4.
 - Secure software can access the Non-Secure view of this register via MPU_MAIR1_NS located at 0xE002_EDC4. The location 0xE002_EDC4 is reserved to software executing in Non-Secure state and the debugger.
 - This register is banked between security states.
- **Preface:** This register is reserved if no MPU regions are implemented in the corresponding security state.

31	30	29	28	27	26	25	24
ATTR7							
R/W							
23	22	21	20	19	18	17	16
ATTR6							
R/W							
15	14	13	12	11	10	9	8
ATTR5							
R/W							
7	6	5	4	3	2	1	0
ATTR4							
R/W							

Bit	Name	Access	Description
31:24	ATTR7	R/W	Memory attributes encoding for MPU regions with an AttrIdx of 7. For the possible values of this field, refer to MAIR_ATTR for encoding. This field resets to an unknown value on a Warm reset.
23:16	ATTR6	R/W	Memory attributes encoding for MPU regions with an AttrIdx of 6. For the possible values of this field, refer to MAIR_ATTR for encoding. This field resets to an unknown value on a Warm reset.
15:8	ATTR5	R/W	Memory attributes encoding for MPU regions with an AttrIdx of 5. For the possible values of this field, refer to MAIR_ATTR for encoding. This field resets to an unknown value on a Warm reset.
7:0	ATTR4	R/W	Memory attributes encoding for MPU regions with an AttrIdx of 4. For the possible values of this field, refer to MAIR_ATTR for encoding. This field resets to an unknown value on a Warm reset.

3.3 Memory Attribute Indirection Register Attributes (MAIR_ATTR)

The MAIR_ATTR characteristics are:

- **Purpose:** Defines the memory attribute encoding for use in the MPU_MAIRO and MPU_MAIR1 register.
- **Usage constraints:** None.
- **Configurations:** All.

- **Attributes:** 8-bit payload.

3.3.1 Outer

When Outer == 0000:

Bit	Name	Access	Description
7:4	Outer	R/W	Outer attributes. Specifies the Outer memory attributes. The possible values of this field are: <ul style="list-style-type: none"> ● 0000: Device memory. ● 00RW: Normal memory, Outer Write-Through transient (RW != 00). ● 0100: Normal memory, Outer Non-Cacheable. ● 01RW: Normal memory, Outer Write-Back Transient (RW != 00). ● 10RW: Normal memory, Outer Write-Through Non-transient. ● 11RW: Normal memory, Outer Write-Back Non-transient. Note: R and W specify the outer read and write allocation policy. <ul style="list-style-type: none"> ● 0: Don't allocate ● 1: Allocate
3:2	Device	R/W	Device attributes. Specifies the memory attributes for device. <ul style="list-style-type: none"> ● 00: Device-nGnRnE ● 01: Device-nGnRE ● 10: Device-nGRE ● 11: Device-GRE
1:0	RSVD	R/W	Reserved

3.3.2 Inner

When Outer != 0000:

Bit	Name	Access	Description
7:4	Outer	R/W	Outer attributes. Specifies the Outer memory attributes. The possible values of this field are: <ul style="list-style-type: none"> ● 0000: Device memory. ● 00RW: Normal memory, Outer Write-Through transient (RW != 00). ● 0100: Normal memory, Outer Non-Cacheable. ● 01RW: Normal memory, Outer Write-Back Transient (RW != 00). ● 10RW: Normal memory, Outer Write-Through Non-transient. ● 11RW: Normal memory, Outer Write-Back Non-transient. Note: R and W specify the outer read and write allocation policy. <ul style="list-style-type: none"> ● 0: Don't allocate ● 1: Allocate
3:0	Inner	R/W	Inner attributes. Specifies the Inner memory attributes. <ul style="list-style-type: none"> ● 0000: Unpredictable. ● 00RW: Normal memory, Inner Write-Through Transient (RW != 00). ● 0100: Normal memory, Inner Non-Cacheable. ● 01RW: Normal memory, Inner Write-Back Transient (RW != 00). ● 10RW: Normal memory, Inner Write-Through Non-transient. ● 11RW: Normal memory, Inner Write-Back Non-transient. Note: R and W specify the outer read and write allocation policy. <ul style="list-style-type: none"> ● 0: Don't allocate ● 1: Allocate

4 Nested Vectored Interrupt Controller (NVIC)

4.1 Features

All interrupts including the core exceptions are managed by the nested vector interrupt controller (NVIC). The NVIC includes the following features:

- The NVIC is an integral part of each CPU.
- Tightly coupled interrupt controller provides low interrupt latency.
- The NVIC controls system exceptions and peripheral interrupts.
- The NVIC of the KM4 supports:
 - 58 vectored interrupts.
 - 8 programmable interrupt priority levels with hardware priority level masking.
 - vector table offset register (VTOR).
- The NVIC of the KM0 supports:
 - 32 vectored interrupts.
 - 4 programmable interrupt priority levels with hardware priority level masking.
 - vector table offset register (VTOR).
- The NVIC supports for NMI from any interrupt.

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

4.2 NVIC Diagram

Most of KM0 interrupt signals are linked to KM4 NVIC at the same interrupt number like LOGUART, and other KM0 interrupt signals are not linked to KM4 NVIC such as WDG, RXI300, and IPC.

This mechanism enables KM4 to use KM0's peripheral as it is KM4's peripheral. But if an interrupt signal is shared by KM0 and KM4, just only one CPU can open this interrupt; if not, software will hang.

The diagram of NVIC is shown in Fig 4-1.

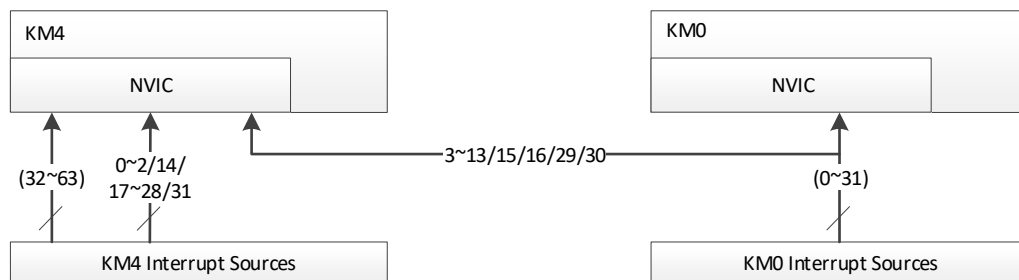


Fig 4-1 NVIC diagram

4.3 NVIC Table

Table 4-1 lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source. The interrupt number doesn't imply any interrupt priority.

Note: The macro "configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY", which is used to define the highest interrupt priority controlled by FreeRTOS, is very important to interrupt. If the interrupt safe FreeRTOS API functions are called by the interrupt service routine, the priority of it must not be higher than this macro. Do not call RTOS_ISR functions from any interrupt that has a higher priority than this macro.

Table 4-1 NVIC table

IRQ	KM4 Name	KM0 Name	Description
	Reset	Reset	Reset
	NMI	NMI	Non-maskable interrupt (external NMI input). The WDG is linked to the NMI vector.
	HardFault	HardFault	All fault conditions if the corresponding fault handler is not enabled.
	MemManagerFault	MemManagerFault	Memory management fault, MPU violation or access to illegal locations
	BusFault	BusFault	Bus error, occurs when Advanced High-Performance Bus (AHB) interface receives an error response from a bus slave (also called pre-fetch abort if it is an instruction fetch, or data abort if it is a data access).
	UsageFault	UsageFault	Exceptions resulting from program error or trying to access coprocessor (the Cortex-M4 does not support a coprocessor).
	RSVD	RSVD	Reserved
	SVC	SVC	Supervisor Call
	Debug Monitor	Debug Monitor	Debug monitor (breakpoints, watchpoints, or external debug requests)
	RSVD	RSVD	Reserved
	PendSV	PendSV	Pendable Service Call
	SYSTICK	SYSTICK	System Tick Timer
0	KM4_System_ISR	KM0_System_ISR	<ul style="list-style-type: none"> ● KM4: System ISR including wakeup event, like HS BT/USB device/SDIO device/UART0 ● KM0: LS peripheral wakeup event, like GPIO/LUART/I2C0, etc.
1	WDG_ISR_H	WDG_ISR_L	KM4 & KM0 watchdog warning interrupt
2	RXI300_IRQ_H	RXI300_IRQ_L	KM4 & KM0 RXI300 platform interrupt
3	UART log	UART log	KM4 & KM0 LP_UART0 (LOGUART) interrupt
4	GPIOA	GPIOA	KM4 & KM0 GPIO PortA Interrupt (GPIO IP0 PORTA)
5	RTC	RTC	KM4 & KM0 RTC interrupt
6	I2C0	I2C0	KM4 & KM0 I2C0 interrupt
7	SPI_FLASH	SPI_FLASH	KM4 & KM0 SPI FLASH interrupt
8	GPIOB	GPIOB	KM4 & KM0 GPIO PortB Interrupt (GPIO IP1 PORTA)
9	LUART	LUART	KM4 & KM0 LP_UART1 interrupt
10	Key-Scan	Key-Scan	KM4 & KM0 Key-Scan interrupt
11	Cap-Touch	Cap-Touch	KM4 & KM0 Cap-Touch interrupt
12	BOR2	BOR2	KM4 & KM0 BOR interrupt
13	SGPIO	SGPIO	KM4 & KM0 SGPIO interrupt
14	IPC_KM0	IPC_KM4	IPC interrupt: <ul style="list-style-type: none"> ● KM0 IPC interrupts KM4 ● KM4 IPC interrupts KM0
15	ADC	ADC	KM4 & KM0 ADC interrupt
16	Q-Decoder	Q-Decoder	KM4 & KM0 Q-Decoder interrupt
17	HTimer0	LTimer0	<ul style="list-style-type: none"> ● KM4 HTimer0 interrupt ● KM0 LTimer0 interrupt
18	HTimer1	LTimer1	<ul style="list-style-type: none"> ● KM4 HTimer1 interrupt ● KM0 LTimer1 interrupt
19	HTimer2	LTimer2	<ul style="list-style-type: none"> ● KM4 HTimer2 interrupt ● KM0 LTimer2 interrupt
20	HTimer3	LTimer3	<ul style="list-style-type: none"> ● KM4 HTimer3 interrupt ● KM0 LTimer3 interrupt
21	HTimer4	LTimer4	<ul style="list-style-type: none"> ● KM4 HTimer4 interrupt ● KM0 LTimer4 interrupt
22	HTimer5	LTimer5	<ul style="list-style-type: none"> ● KM4 HTimer5 interrupt ● KM0 LTimer5 interrupt
23	LCDC	LGDMA0_Channel0	<ul style="list-style-type: none"> ● KM4 LCDC interrupt ● KM0 LGDMA0_Channel0 interrupt
24	USB	LGDMA0_Channel1	<ul style="list-style-type: none"> ● KM4 USB interrupt ● KM0 LGDMA0_Channel1 interrupt
25	SDIO_DEV	LGDMA0_Channel2	<ul style="list-style-type: none"> ● KM4 SDIO_DEV interrupt

			<ul style="list-style-type: none"> ● KM0 LGDMA0_Channel2 interrupt
26	SD_HOST	Wi-Fi FISR (0x134) + FESR (0x124)	<ul style="list-style-type: none"> ● KM4 SD_HOST interrupt ● KM0 Wi-Fi FISR + FESR interrupt
27	IPSEC	Wi-Fi FTSR (0x13C) + mailbox	<ul style="list-style-type: none"> ● KM4 SD_HOST interrupt ● KM0 Wi-Fi FTSR + mailbox interrupt
28	I2S0	LGDMA0_Channel3	<ul style="list-style-type: none"> ● KM4 I2S0 interrupt ● KM0 LGDMA0_Channel3 interrupt
29	PWR_DOWN	PWR_DOWN	KM4 & KM0 PWR_DOWN interrupt. This interrupt will be triggered when power down pin pushes under power down interrupt mode.
30	ADC_COMP	ADC_COMP	KM4 & KM0 ADC comparator interrupt
31	WL_DMA	KM4_WAKE_ISR	<ul style="list-style-type: none"> ● KM4 WL_DMA interrupt ● KM0: KM4 peripherals wake interrupt, the same as KM4_System_ISR
32	WL_PROTOCOL (0xB4)	N/A	KM4 WL_PROTOCOL interrupt
33	PSRAMC	N/A	KM4 PSRAMC interrupt
34	UART0	N/A	KM4 HS_UART0 interrupt
35	UART1_BT	N/A	KM4 HS_UART1_BT interrupt
36	SPI0	N/A	KM4 HS_SPI0 interrupt
37	SPI1	N/A	KM4 HS_SPI1 interrupt
38	HUSIO	N/A	KM4 HS_USIO interrupt
39	IR	N/A	KM4 IR interrupt
40	BT_SCORE_BOARD	N/A	KM4 BT_SCORE_BOARD interrupt
41	GDMA0_Channel0	N/A	KM4 HS_GDMA0_Channel0 interrupt
42	GDMA0_Channel1	N/A	KM4 HS_GDMA0_Channel1 interrupt
43	GDMA0_Channel2	N/A	KM4 HS_GDMA0_Channel2 interrupt
44	GDMA0_Channel3	N/A	KM4 HS_GDMA0_Channel3 interrupt
45	GDMA0_Channel4	N/A	KM4 HS_GDMA0_Channel4 interrupt
46	GDMA0_Channel5	N/A	KM4 HS_GDMA0_Channel5 interrupt
47	RSVD	N/A	Reserved
48	RSVD	N/A	Reserved
49	RSVD	N/A	Reserved
50	IPSEC_S	N/A	KM4 HS_IPSEC TrustZone interrupt
51	RXI300_IRQ_S	N/A	KM4 HS_RXI300 TrustZone interrupt
52	GDMA0_Channel0_S	N/A	KM4 HS_GDMA0_Channel0 TrustZone interrupt
53	GDMA0_Channel1_S	N/A	KM4 HS_GDMA0_Channel1 TrustZone interrupt
54	GDMA0_Channel2_S	N/A	KM4 HS_GDMA0_Channel2 TrustZone interrupt
55	GDMA0_Channel3_S	N/A	KM4 HS_GDMA0_Channel3 TrustZone interrupt
56	GDMA0_Channel4_S	N/A	KM4 HS_GDMA0_Channel4 TrustZone interrupt
57	GDMA0_Channel5_S	N/A	KM4 HS_GDMA0_Channel5 TrustZone interrupt

4.4 NVIC Register Description

The physical base address of NVIC is 0xE000_E000. Table 4-2 listed the details of the NVIC registers.

Table 4-2 Memory map of NVIC

Name	Offset	Access	Reset	Description
ISER0	0x100	R/W	0	Interrupt Set Enable Register 0. This register allows enabling interrupts and reading back the interrupt enabled state for peripheral functions.
ISER1	0x104	R/W	0	Interrupt Set Enable Register 1. See ISER0 description.
ICER0	0x180	R/W	0	Interrupt Clear Enable Register 0. This register allows disabling interrupts and reading back the interrupt enabled state for peripheral functions.
ICER1	0x184	R/W	0	Interrupt Clear Enable Register 1. See ICER0 description.

ISPR0	0x200	R/W	0	Interrupt Set Pending Register 0. This register allows changing the interrupt state to pending and reading back the interrupt pending state for peripheral functions.
ISPR1	0x204	R/W	0	Interrupt Set Pending Register 1. See ISPR0 description.
ICPR0	0x280	R/W	0	Interrupt Clear Pending Register 0. This register allows changing the interrupt state to no pending and reading back the interrupt pending state for peripheral functions.
ICPR1	0x284	R/W	0	Interrupt Clear Pending Register 1. See ICPR0 description.
IABR0	0x300	RO	0	Interrupt Active Bit Register 0. This register allows reading the current interrupt active state for specific peripheral functions.
IABR1	0x304	RO	0	Interrupt Active Bit Register 1. See IABR0 description.
IPR0	0x400	R/W	0	Interrupt Priority Register 0. This register contains the 3-bit priority fields for interrupt 0 to 3. ()
IPR1	0x404	R/W	0	Interrupt Priority Register 1. This register contains the 3-bit priority fields for interrupt 4 to 7.
IPR2	0x408	R/W	0	Interrupt Priority Register 2. This register contains the 3-bit priority fields for interrupt 8 to 11.
IPR3	0x40C	R/W	0	Interrupt Priority Register 3. This register contains the 3-bit priority fields for interrupt 12 to 15.
IPR4	0x410	R/W	0	Interrupt Priority Register 4. This register contains the 3-bit priority fields for interrupt 16 to 19.
IPR5	0x414	R/W	0	Interrupt Priority Register 5. This register contains the 3-bit priority fields for interrupt 20 to 23.
IPR6	0x418	R/W	0	Interrupt Priority Register 6. This register contains the 3-bit priority fields for interrupt 24 to 27.
IPR7	0x41C	R/W	0	Interrupt Priority Register 7. This register contains the 3-bit priority fields for interrupt 28 to 31.
IPR8	0x420	R/W	0	Interrupt Priority Register 8. This register contains the 3-bit priority fields for interrupt 32 to 35.
IPR9	0x424	R/W	0	Interrupt Priority Register 9. This register contains the 3-bit priority fields for interrupt 36 to 39.
IPR10	0x428	R/W	0	Interrupt Priority Register 10. This register contains the 3-bit priority field for interrupt 40 to 43.
IPR11	0x42C	R/W	0	Interrupt Priority Register 11. This register contains the 3-bit priority fields for interrupt 44 to 47.
IPR12	0x430	R/W	0	Interrupt Priority Register 12. This register contains the 3-bit priority fields for interrupt 48 to 51.
IPR13	0x434	R/W	0	Interrupt Priority Register 13. This register contains the 3-bit priority fields for interrupt 52 to 55.
IPR14	0x438	R/W	0	Interrupt Priority Register 14. This register contains the 3-bit priority fields for interrupt 56 to 58.
STIR	0xF00	WO	-	Software Trigger Interrupt Register, allows software to generate interrupts.

Note: For KM0, IPR registers contain the 2-bit priority fields for each interrupt.

4.4.1 ISER0 and ISER1

The ISER0 register allows enabling the first 32 peripheral interrupts, or reading the enabled state of those interrupts. The remaining interrupts are enabled via the ISER1 register. The interrupts can be disabled through the ICER0 and ICER1 registers.

Each bit in the ISER0 and ISER1 registers controls one interrupt in NVIC table.

The operation of read/write to this register means that:

- Write: Writing 0 has no effect, writing 1 enables the interrupt.
- Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

4.4.2 ICER0 and ICER1

The ICER0 register allows disabling the first 32 peripheral interrupts, or reading the enabled state of those interrupts. The remaining interrupts are disabled via the ICER1 register. The interrupts can be enabled through the ISER0 and ISER1 registers.

The operation of read/write to this register means that:

- Write: Writing 0 has no effect, writing 1 disables the interrupt.
- Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

4.4.3 ISPR0 and ISPR1

The ISPR0 register allows setting the pending state of the first 32 peripheral interrupts, or reading the pending state of those interrupts. The remaining interrupts can have their pending state set via the ISPR1 register. The pending state of the interrupts can be cleared through the ICPR0 and ICPR1 registers.

The operation of read/write to this register means that:

- Write: Writing 0 has no effect, writing 1 change the interrupt state to pending.
- Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.

4.4.4 ICPR0 and ICPR1

The ICPR0 register allows clearing the pending state of the first 32 peripheral interrupts, or reading the pending state of those interrupts. The remaining interrupts can have their pending state cleared via the ICPR1 register. The pending state of the interrupts can be set through the ISPR0 and ISPR1 registers.

The operation of read/write to this register means that:

- Write: Writing 0 has no effect, writing 1 changes the interrupt state to not pending.
- Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.

4.4.5 IABR0 and IABR1

The IABR0 register is a read-only register that allows reading the active state of the first 32 peripheral interrupts. Bits in IABR registers are set while the corresponding interrupt service routines are in progress. Additional interrupts active state can be read via the IABR1 register.

The operation of read to this register means that:

- Read: 0 indicates that the interrupt is not active, 1 indicates that the interrupt is active.

4.4.6 IPR0 ~ IPR14

Each IPR register controls the priority of the 4 peripheral interrupts.

- For KM0, each interrupt can have one of 4 priorities (0, 1, 2, 3), where 0 is the highest priority.
- For KM4, each interrupt can have one of 8 priorities (0, 1, 2 ..., 7), where 0 is the highest priority.

4.4.7 STIR

The Software Trigger Interrupt Register (STIR) provides an alternate way for software to generate an interrupt, in addition to using the ISPR registers. This mechanism can only be used to generate peripheral interrupts, not system exceptions.

By default, only privileged software can write to the STIR. Unprivileged software can be given this ability if privileged software sets the USERSETMPEND bit in the CCR register.

The interrupt number to be programmed in the STIR is listed in Table 4-3.

Table 4-3 Software trigger interrupt register

Address Offset	Bit	Name	Description
0xE000_EF00	31:9	RSVD	Reserved. Read value is undefined, only 0 should be written.
	8:0	INTID	Writing a value to this field generates an interrupt for the specified interrupt number.

5 CPU System Tick (SysTick) Timer

The System Tick (SysTick) Timer is present on both the KM4 and the KM0. The SysTick clock is fixed to half the frequency of the system clock.

5.1 Features

- Simple 24-bit timer
- Dedicated exception vector
- Clocked internally by the system clock or the system tick clock (SYSTICKCLK)

5.2 Functional Description

The SysTick timer is an integral part of both the KM4 and the KM0. It is a 24-bit timer that counts down to zero and generates an interrupt, to provide a fixed 1 millisecond time interval between interrupts for use by an operating system or other system management software. The SysTick timer is clocked from the CPU clock (the system clock) or from the SDM32.768 clock. In order to generate recurring interrupts at a specific interval, the SYST_RVR register must be initialized with the correct value for the desired interval.

Since the SysTick timer is a part of the CPU, it facilitates porting of software by providing a standard timer that is available on ARM Cortex-M based devices. The SysTick timer can be used for:

- A RTOS tick timer which fires at a programmable rate (for example 1000Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the core clock.
- A simple counter. Software can use this to measure time completed and used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit in the SYST_CSR register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

Refer to the appropriate ARM Cortex User Guide for more details.

5.3 Register Description

The SysTick timer registers are located on the private peripheral bus of each CPU, and the details of registers are shown in Table 5-1. The physical base address is 0xE000_E000.

Table 5-1 Memory map of SysTick timer

Name	Offset	Access	Description
SYST_CSR	0x010	R/W	System Timer Control and Status Register
SYST_RVR	0x014	R/W	System Timer Reload Value Register
SYST_CVR	0x018	R/W	System Timer Current Value Register
SYST_CALIB	0x01C	RO	System Timer Calibration Value register

5.3.1 SYST_CSR

- **Name:** System Timer Control and Status Register
- **Size:** 32 bits
- **Address offset:** 0x010
- **Read/write access:** read/write

This register contains control information for the SysTick timer and provides a status flag. It is a part of the CPU, and determines the clock source for the SysTick timer.

31	30	29	28	27	...	22	21	20	19	18	17	16
RSVD												COUNTFLAG
												R/W

15	14	13	12	...	7	6	5	4	3	2	1	0
RSVD										CLKSOURCE	TICKINT	ENABLE
										R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:17	RSVD	N/A	-	Reserved. Read value is undefined, only zero should be written.
16	COUNTFLAG	R/W	0	Returns 1 if the SysTick timer counted to 0 since the last read of this register.
15:3	RSVD	N/A	-	Reserved. Read value is undefined, only zero should be written.
2	CLKSOURCE	R/W	0	SysTick clock source selection. <ul style="list-style-type: none"> 1: The system clock is selected. 0: The output clock from the SysTick clock divider (SYSTICKCLKDIV) is selected as the reference clock. Note: When the SysTick clock divider is selected as the clock source, the CPU clock must be at least 2.5 times faster than the divider output.
1	TICKINT	R/W	0	SysTick interrupt enable. <ul style="list-style-type: none"> 1: The SysTick interrupt is enabled. When enabled, the interrupt is generated when the SysTick counter counts down to 0. 0: The SysTick interrupt is disabled.
0	ENABLE	R/W	0	SysTick counter enable. <ul style="list-style-type: none"> 1: The counter is enabled. 0: The counter is disabled.

5.3.2 SYST_RVR

- **Name:** System Timer Reload Value Register
- **Size:** 32 bits
- **Address offset:** 0x014
- **Read/write access:** read/write

This register is set to the value which is loaded into the SysTick timer, whenever it counts down to zero. This register is loaded by software as part of timer initialization. The SYST_CALIB register may be read and used as the value for SYST_RVR register if the CPU is running at the frequency intended for use with the SYST_CALIB value.

31	30	29	...	26	25	24	23	22	21	20	19	...	4	3	2	1	0
RSVD							RELOAD										
							R/W										

Bit	Name	Access	Reset	Description
31:24	RSVD	N/A	-	Reserved. Read value is undefined, only zero should be written.
23:0	RELOAD	R/W	0	The value which is loaded into the SysTick counter when it counts down to 0.

5.3.3 SYST_CVR

- **Name:** System Timer Current Value Register
- **Size:** 32 bits
- **Address offset:** 0x018
- **Read/write access:** read/write

This register returns the current count from the SysTick counter when it is read by software.

31	30	29	...	26	25	24	23	22	21	20	19	...	4	3	2	1	0
RSVD							CURRENT										
							R/W										

Bit	Name	Access	Reset	Description
31:24	RSVD	N/A	-	Reserved. Read value is undefined, only zero should be written.

23:0	CURRENT	R/W	0	Reading this field returns the current value of the SysTick counter. Writing any value.
------	---------	-----	---	---

5.3.4 SYST_CALIB

- **Name:** System Timer Calibration Value Register
- **Size:** 32 bits
- **Address offset:** 0x01C
- **Read/write access:** read-only

This register is read-only and its value is provided by the value of the SYST_CALIB register in the system configuration (SYSCON) block.

31	30	29	28	27	26	25	24	23	22	21	...	2	1	0
NOREF	SKEW	RSVD							TENMS					
R	R								R					

Bit	Name	Access	Reset	Description
31	NOREF	R	0	Indicates whether an external reference clock is available. This bit is loaded from the SYST_CALIB register in SYSCON. <ul style="list-style-type: none"> ● 0: A separate reference clock is available. ● 1: A separate reference clock is not available.
30	SKEW	R	0	Indicates whether the TENMS value generates a precise 10 millisecond time, or an 0 approximation. This bit is loaded from the SYST_CALIB register in SYSCON. <ul style="list-style-type: none"> ● 0: The value of TENMS is considered to be precise. ● 1: The value of TENMS is not considered to be precise.
29:24	RSVD	N/A	-	Reserved. Read value is undefined, only zero should be written.
23:0	TENMS	R	0	This field is loaded 0 from the SYST_CALIB register in SYSCON, and always RAZ/WI (read as zero, write ignored).

6 Pad Control and Pinmux

6.1 Features

The following electrical properties are configurable for standard I/O pins:

- Function ID
- Pull-up/pull-down resistor
- Schmitt trigger
- Pad driving strength
- Pad shut down
- I²C open drain mode

6.2 Functional Description

The PADCTRL register controls the function of the device pins. Each GPIO pin has a dedicated control register to select its function and characteristics. Each I/O pin has a unique set of functional capabilities. Not all pin characteristics are selectable on each pin. For instance, pins that have an I²C function can be configured for different I²C bus modes.

The pad diagram is given in Fig 6-1. There are many kinds of pads in Ameba-D, different pads have different configuration. For example, normal pad and I²C pad have different pull resistance.

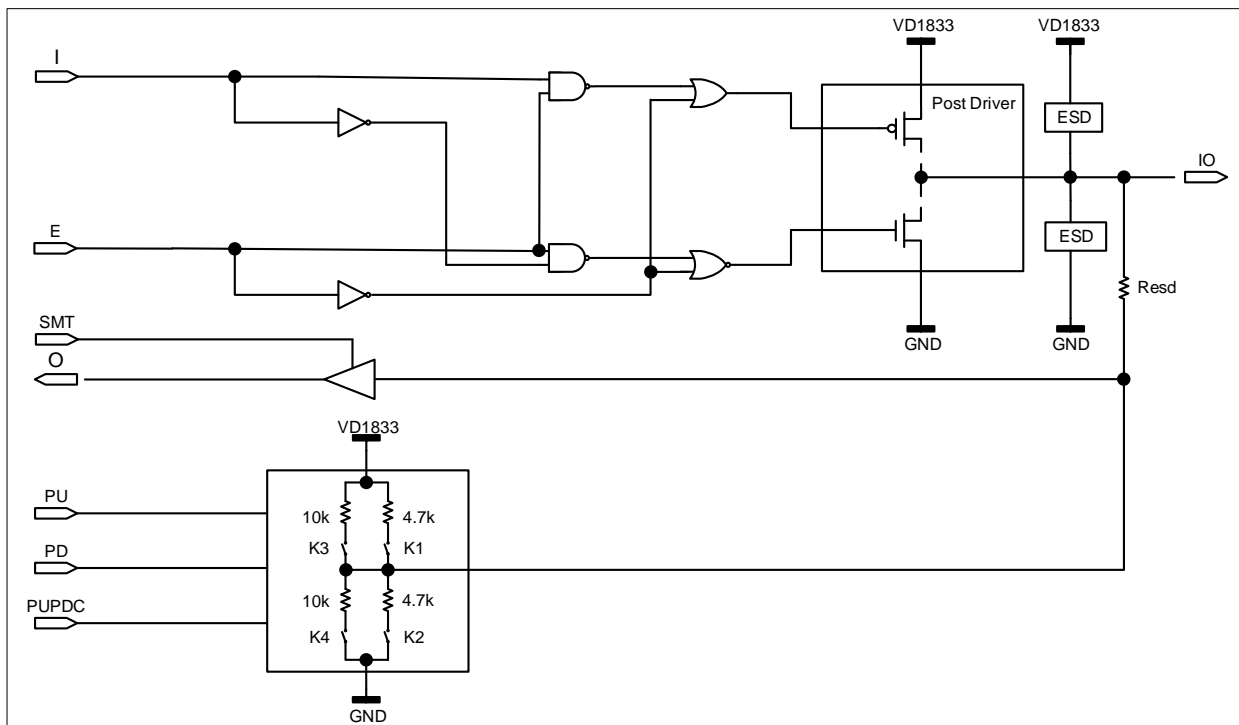


Fig 6-1 Pad diagram

6.2.1 Pad Types

The driving capability and internal pull up/down resistor of different pads are listed in Table 6-1.

Table 6-1 Pad types

Pad type	Loading (pF) ¹	Driving (mA)		Internal pull resistor (ohm)	Pin name		Max. clock rate	Resistor available in deepsleep mode ³
		1.8V	3.3V		PA	PB		
Normal	15	2/4/6/8mA	4/8/12/16mA	50k	PA[7], PA[8], PA[9], PA[10], PA[11], PA[22], PA[23], PA[24], PA[27], PA[28]	PB[8], PB[9], PB[10], PB[11], PB[12], PB[13], PB[14], PB[15], PB[16], PB[17]	100MHz	Yes
SDIO	15	2/4/6/8/10/12mA	4/8/12/16/20/24mA	50k	PB[18], PB[19], PB[20], PB[21], PB[22], PB[23], PB[24], PB[25]		200MHz	Yes
Key-Scan	15	4/8mA	8/16mA	4.7k/50k	PA[12], PA[13], PA[14], PA[15], PA[16], PA[17], PA[18], PA[19], PA[20], PA[21], PA[25], PA[26]		50MHz	Yes
Audio	15	2/4/6/8mA	4/8/12/16mA	PU 29k PL 26k	PA[0], PA[1], PA[2], PA[3], PA[4], PA[5], PA[6]	PB[28], PB[29], PB[30], PB[31]	25MHz	No
Cap-Touch ²	< 2	2/4mA	4/8mA	50k		PB[4], PB[5], PB[6], PB[7]	25MHz	Yes
I ² C	15	4/8mA	8/16mA	4.7k/10k	PA[29], PA[30], PA[31]	(I ² C) PB[0], PB[26], PB[27] (ADC) PB[1], PB[2], PB[3]	50MHz	Yes
5V Tolerate	15	2/4mA	4/8mA	50k	VBAT			

- This field means that the external capacitance of the pins cannot exceed the value of loading (pF) when the pad works in the max. output clock.
- For this pad type:
 - When used as ordinary GPIO pad, make sure the loading capacitance less than 15pF.
 - When used as Cap-Touch pad, keep the loading capacitance as small as possible for better sensitivity.
- In deepsleep mode, the PA[0] ~ PA[6] and PB[28] ~ PB[31] pins are in FLOATING state, internal resistors of these pins are NOT available. If circuit connected with these GPIOs needs to be pulled to high or low state, external resistor on PCB is needed. In other mode except deepsleep or sleep mode, internal resistors of all GPIOs are available.

6.2.2 Pad Pull Resistor Control

The PAD_BIT_PULL_DOWN_RESISTOR_EN and PAD_BIT_PULL_UP_RESISTOR_EN bits in the PADCTRL register allow the selection of on-chip pull-up or pull-down resistors for each pin.

The possible on-chip resistor configurations are pull-up enabled, pull-down enabled, or no pull-up/pull-down, as shown in Table 6-2. Pull-down and pull-up resistor's value is different between different pad types.

Table 6-2 I²C pad pull resistor control

Control register	K1	K2	K3	K4	Pull state
PUPDC = x and PU = 0 and PD = 0	Off	Off	Off	Off	No pull
PUPDC = 0 and PU = 1	Off	Off	On	Off	10K pull-up
PUPDC = 1 and PU = 1	On	Off	Off	Off	4.7K pull-up
PUPDC = 0 and PD = 1	Off	Off	Off	On	10K pull-down
PUPDC = 1 and PD = 1	Off	On	Off	Off	4.7K pull-down
PUPDC = x and PU = 1 and PD = 1	The setting is not allowed				

6.2.3 Pad Schmitt Trigger

All I/O pins include a schmitt trigger that can be selectively disabled by setting the PAD_BIT_SCHMITT_TRIGGER_EN bit in the PADCTRL register. The specification of schmitt trigger is listed in Table 6-3.

Table 6-3 Schmitt trigger specification

I/O power	1.8V	3.3V
Schmitt trigger	VIH: (0.65*VDD) V	VIH: 2V
	VIL: (0.35*VDD) V	VIL: 0.8V

6.2.4 Pad Driving Strength

The I/O pad driving strength can be configured through PAD_BIT_DRIVING_STRENGTH field in the PADCTRL register, as shown in Table 6-4. Pad driving strength is different between different pad types.

Table 6-4 Normal pad driving strength configuration

Pad driving strength	1.8V	3.3V
0	2mA	4mA
1	4mA	8mA
2	6mA	12mA
3	8mA	16mA

6.2.5 Pad Shutdown

The power of a I/O pad can be shut down through the PAD_BIT_SHUT_DWON bit in the PADCTRL register. You can use this function to conserve power.

6.2.6 I²C Open-drain Mode

Pins that support I²C with specialized pad electronics have additional configuration bits. These bits have multiple configurations to support I²C variants. They are not hard-wired so that the pins can be more easily used for non-I²C function.

In Fig 6-1, signal “E” and signal “I” are used to select push-pull mode or open-drain mode, the two signals are controlled by hardware automatically. For example, when you select I²C function for this pin, open-drain mode is selected.

Post driver output mode	E	I	IO
Normal GPIO	1	Driven by GPIO module	Push-pull output
I ² C	Driven by I ² C module	0	Open-drain output

Pay attention that PB[5] and PB[6] can also be set to I²C function, but their pull resistors only support 50Kohm. So 4.7kohm external pull-up resistors should be added to ensure I²C work properly.

6.2.7 Audio Pad

As Fig 6-1Fig 6-2 shows, the circuit of audio pad for PA[0] ~ PA[6] is different from that of PB[28] ~ PB[31]. In the circuit, the LS module is used to level shift, which can switch the level from 0.9V to 3.3V.

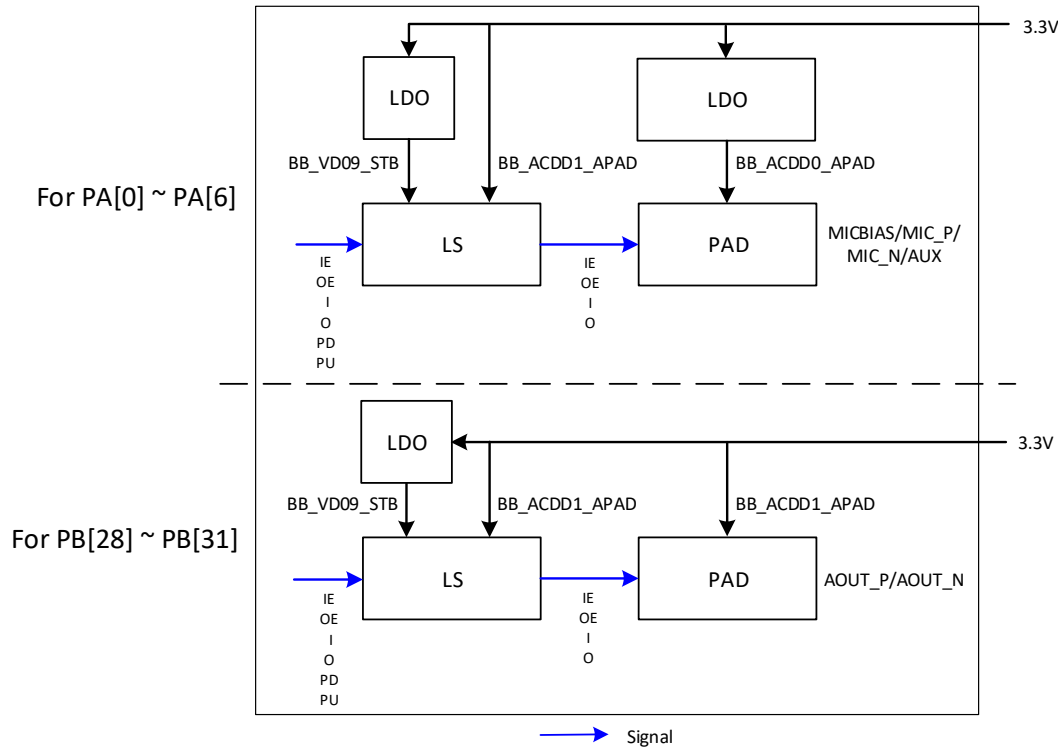


Fig 6-2 Circuit diagram of audio pad

The I/O power of GPIO pins is from pin AVCC, which is output of internal LDO. The GPIOs of PA[0] ~ PA[6] have some restriction because of pinmux with audio function. Take care when using these pins as other function except audio.

- When the audio function is used, the maximum voltage of these GPIOs can reach to 2.8V due to 3.3V powered on, and reach to 1.6V due to 1.8V powered on.
- When the audio function is not used, the maximum voltage of these GPIOs can reach to the supply voltage (3.3V or 1.8V).
- If you want to use these GPIOs in sleep mode, you should enable audio LDO. And the background current will increase from 21uA to 230uA.

The operation of using audio pins as normal GPIOs is illustrated in Table 6-5.

Table 6-5 Using audio pins as normal GPIOs

Pin name	Operations
PA[0] ~ PA[4]	<ul style="list-style-type: none"> ● Enable audio LDO ● Modify audio LDO voltage setting (0x4800_0344[7:1])
PA[5], PA[6]	<ul style="list-style-type: none"> ● Enable audio LDO ● Modify audio LDO voltage setting (0x4800_0344[7:1]) ● Mute audio LINE IN

Note: For A-Cut, the maximum AVCC can only reach to 3.064V. For B-Cut/C-Cut, this issue is fixed and AVCC can reach to 3.3V. The method to configure audio LDO voltage is illustrated in Table 6-6.

Table 6-6 Audio LDO voltage setting

0x4800_0344[7:1]	AVCC	
	A-Cut	B-Cut/C-Cut
0x20	2.8V	-
0x00	3.064V	-
0x2A	-	2.8V
0x00	-	3.3V

As PA[5] and PA[6] are sharing with audio LINE IN, there exists a 30kohm pull-down resistor to GND in effect. When used as GPIO, you need to disable the pull-down by muting audio LINE IN. The method is listed in Table 6-7.

Table 6-7 Mute configuration

Mute control	Register configuration
Mute MIC IN	Sets Audio Codec Register 0x03 bit[4], bit[6] to enable MIC IN mute by setting register 0x40010000 to 0x08500301
Mute LINE IN	Sets Audio Codec Register 0x03 bit[5], bit[7] to enable LINE IN mute by setting register 0x40010000 to 0x08A00301
Mute MIC IN and LINE IN	Sets Audio Codec Register 0x03 bit[7:4] to enable MIC IN and LINE IN mute by setting register 0x40010000 to 0x08F00301

For the QFN88 Story Machine package, in the case of both audio and SD function are used, PA[5], PA[6] are used as SD_WP/SD_CD. As the AVCC is 2.8V, to detect the card's inserted/removed/protected state effectively, it is suggested to add 4.7Kohm pull-up resistor to AVCC.

6.3 Pin Multiplexing Function

The PAD_BIT_FUNCTION_ID field in the PADCTRL register can be set to GPIO (typically value 000) or a special function.

- For pins set to GPIO, the GPIO IP registers determine whether the pin is configured as an input or an output.
- For any special function, the pin direction is controlled automatically depending on the function.

The microcontroller I/O pins are connected to onboard peripherals/modules through a multiplexer that allows only one peripheral's function connected to an I/O pin at a time. In this way, there can be no conflict between peripherals sharing the same I/O pin.

Each I/O pin has a multiplexer with up to 32 alternate function inputs (AF0 to AF31) that can be configured through the PADCTRL register.

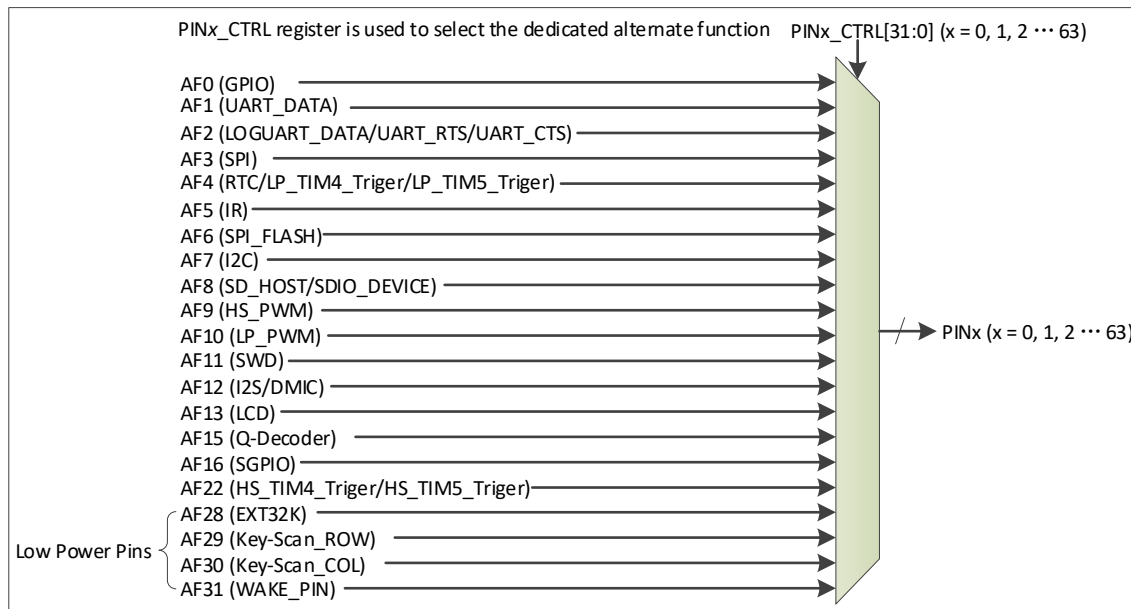


Fig 6-3 Selecting an alternate function on Ameba-D

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate function mapped on different I/O pins to optimize the number of peripherals available in smaller package. Refer to the UM0402 pinmux table for detailed mapping of the system and peripherals' alternate function I/O pins.

Note:

In the process of system power on, the state of GPIOs is described below.

- AON GPIO:

- When pressing the **Reset** button, all the GPIOs can keep the original state as before; while releasing the **Reset** button, the state of GPIOs are similar with the state in the process of system power on described above.

Each I/O pin has one PADCTRL register assigned to control the pin's electrical characteristics.

31		30		29		...		18		17		16			
RSVD															
15		14		13		12		11		10		9		8	
PAD_BIT_SHUT_DOWN		PAD_BIT_SDIO_H3L1		RSVD		PAD_BIT_SCHMITT_TRIGGER_EN		PAD_BIT_DRIVING_STRENGTH		PAD_BIT_PULL_DOWN_RESISTOR_EN		PAD_BIT_PULL_UP_RESISTOR_EN			
R/W		R/W				R/W		R/W		R/W		R/W			
7		6		5		4		3		2		1		0	
RSVD						PAD_BIT_FUNCTION_ID									
						R/W									

Address	Bit	Name	Access	Description
0x4800_0400 ~ 0x4800_04FC	31:16	RSVD	N/A	Reserved
	15	PAD_BIT_SHUT_DWON	R/W	Pad shutdown enable
	14	PAD_BIT_SDIO_H3L1	R/W	Pad I/O voltage selection for SDIO pad. <ul style="list-style-type: none"> 1: 3.3V 0: 1.8V
	13	RSVD	N/A	Reserved
	12	PAD_BIT_SCHMITT_TRIGGER_EN	R/W	Controls “SMT” signal, used to enable schmitt trigger.
	11:10	PAD_BIT_DRIVING_STRENGTH	R/W	<ul style="list-style-type: none"> For PDA, drives pad {bit[11], bit[10]} to control pad driving strength. <ul style="list-style-type: none"> 00: 2mA 01: 4mA 10: 8mA 11: 16mA For PDC/F/G, drives pad {bit[11]} to control pull resistor value and {bit[10]} to control driving strength respectively. <ul style="list-style-type: none"> 1: Select the small resistance 0: Select the small driving strength For PDD/E, drives pad {bit[10]} to control driving strength.
	9	PAD_BIT_PULL_DOWN_RESISTOR_EN	R/W	Controls “PD” signal, pull-down resistor enable, and resistor is 50Kohm.
	8	PAD_BIT_PULL_UP_RESISTOR_EN	R/W	Controls “PU” signal, pull-up resistor enable, and resistor is 50Kohm.
	7:5	RSVD	N/A	Reserved for function ID extension
	4:0	PAD_BIT_FUNCTION_ID	R/W	Function ID

7 Inter Processor Communication (IPC)

There are two CPUs, High Power (HP) CPU and Low Power (LP) CPU, on the Ameba-D platform. The inter processor communication (IPC) hardware is designed to make two CPUs communicate with each other.

The IPC module has 32 interrupt sources. Each source has a set of enable, disable, clear, set, mask and status registers. The interrupt sources are ORED, and the IPC interrupt output line is connected to the interrupt controller input.

7.1 Features

- Allowing multiple CPUs to share resources and communicate with each other in a simple manner
- Up to 32 user-defined interrupts to its partner of each CPU
- 16 hardware semaphores for claim shared resources if it is available
- Dummy registers for communication between KM4 and KM0

7.2 Functional Description

7.2.1 Architecture Block Diagram

The IPC provides a means for simple communication between CPUs. The system architecture is shown in Fig 7-1.

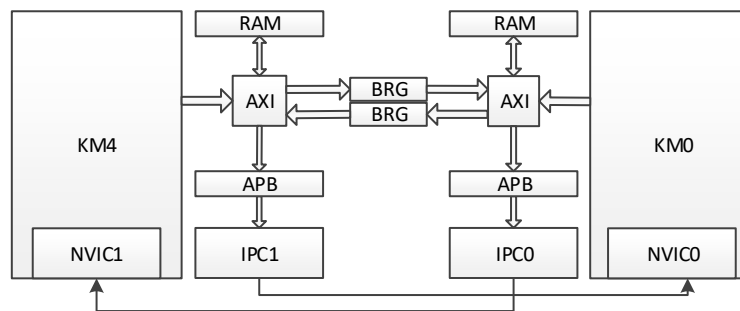


Fig 7-1 IPC system architecture

7.2.2 Core-to-Core Interrupt

Each CPU can cause up to 32 user-defined interrupts to another core. So, there are five 32-bit registers for each processor.

- IPCx_IER/IPCx_IDR
- IPCx_IRR/IPCx_ICR
- IPCx_ISR

Each of the 32 bits of these registers represents 32 independent channels through which the KM4 CPU can send up to 32 events to the KM0 CPU via software handshaking.

- Writing a '1' to a bit of IPC1_IRR fires an interrupt into direction KM0, writing to the corresponding bit in IPC1_ICR clears this interrupt.
- Writing a '1' to a bit of IPC0_IRR fires an interrupt into direction KM4, writing to the corresponding bit in IPC0_ICR clears this interrupt.

The interrupt request of IPC is shown in Fig 7-2. For example, to send an event via channel 2 from KM4 to KM0, the KM4 and KM0 CPUs use bit[2] of the IPC1_IRR/IPC1_ICR/IPC1_ISR registers. The handshake starts with the KM4 polling bit[2] of the IPC1_ISR register to make sure bit[2] is '0'. Next, the KM4 writes a '1' to bit[2] of the IPC1_IRR register to start the handshake. As soon as the KM4 writes a '1' to bit[2] of the IPC1_IRR register, bit[2] of IPC1_ISR also turns to '1', thus announcing the IPC1_IRQ2 interrupt to the KM0. As the KM0 reads a '1' from the IPC1_ISR register, the KM0 CPU is acknowledged by writing a '1' to bit[2] of the IPC1_ICR register, which in turn, clears bit[2] of the IPC1_ISR register, enables KM4 to send another message.

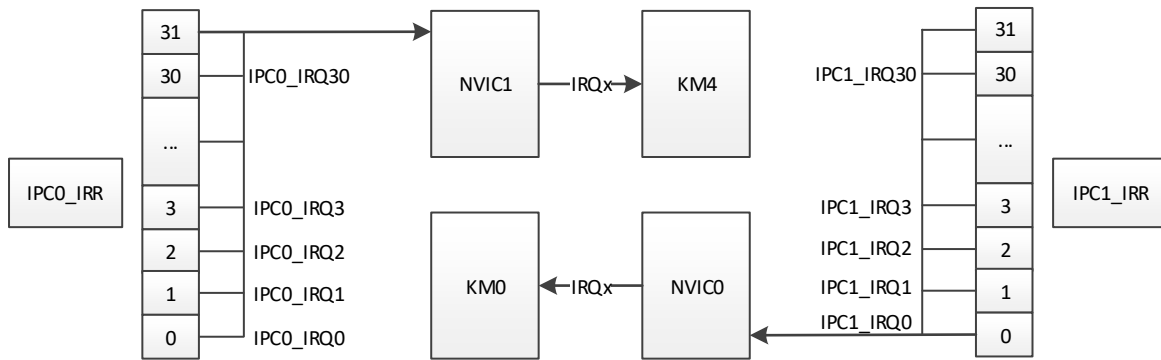


Fig 7-2 IPC interrupt request

7.2.3 Hardware Semaphore

The hardware semaphore is used for a mutual exclusion mechanism. It can be accessed by both cores in atomic operation. When reading for any reason, the current value is returned and the bit is cleared. The bit is set again following any write.

As Fig 7-3 shows, each hardware semaphore uses 2 bits of an IPCO_SEM register. One bit is named as 'CPUID', which is used to indicate who owns this semaphore. Another bit is named as 'Free', which is used to indicate the status of the semaphore – free or occupied.

'CPUID' bit is a read-only bit, it is set automatically by hardware when 'Free' bit is set.

'Free' bit is a R/W bit, and writing a '0' has not any effect on this register. The following is the specification for this bit:

- 0: When writing 1, hardware sets CPUID.
- 1: When writing 1, hardware clears 'CPUID' bit and 'Free' bit, and just owner CPU can clear these bits.

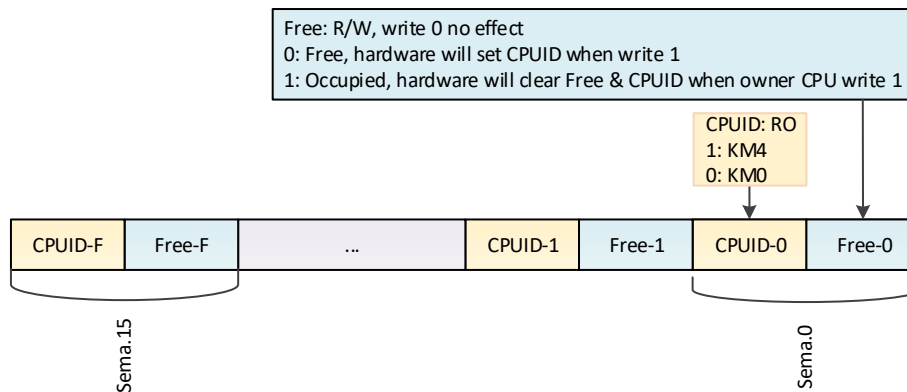


Fig 7-3 IPC hardware semaphore mechanism

For example, The KM4 wants to access a shared ring buffer structure, it takes the following steps:

- (1) KM4 reads the IPCO_SEM[0] to check the 'Free' bit of this semaphore. If this semaphore is occupied ('Free' bit = 1), means the corresponding resource is in using. The software needs to wait this semaphore to be free. If the semaphore is free ('Free' bit = 0), then go to the next step.
- (2) KM4 writes a '1' to IPCO_SEM[0], hardware helps to set the CPUID if writing is success for this CPU.
- (3) KM4 reads the IPCO_SEM[1] to check if the 'CPUID' bit is equal to its CPUID. If matched, means it has acquired this semaphore successfully (another CPU is forbidden to clear this bit). Otherwise, the semaphore acquisition is failed. It may be caused by another CPU which also tries to acquire this semaphore at the same time and this semaphore has been gotten by another CPU.
- (4) KM4 accesses the shared ring buffer structure.
- (5) KM4 writes a '1' to IPCO_SEM[0], hardware helps to clear the CPUID and 'Free' bit, so that another CPU can get this semaphore if needed. you can give interrupt (IPCx_IRR) to peer CPU for the semaphore free.

7.3 IPC Registers

The details of IPC control registers are listed in Table 7-1.

Table 7-1 IPC control registers

Name	Address Offset	Access	KM0/KM4	Description
IPCx_IER	0x0000	WO	KM0/KM4	Interrupt Enable Register <ul style="list-style-type: none"> 1: Writing a '1' to a bit of this register enables the corresponding interrupt. 0: Writing a '0' has no effect.
IPCx_IDR	0x0004	WO	KM0/KM4	Interrupt Disable Register <ul style="list-style-type: none"> 1: Writing a '1' to a bit of this register disables the corresponding interrupt. 0: Writing a '0' has no effect.
IPCx_IRR	0x0008	WO	KM0/KM4	Interrupt Request Register <ul style="list-style-type: none"> 1: Writing a '1' to a bit of this register sets the corresponding bit in IPCx_ISR, and fires an interrupt into another processor. 0: Writing a '0' has no effect.
IPCx_ICR	0x000C	WO	KM0/KM4	Interrupt Clear Register <ul style="list-style-type: none"> 1: Writing a '1' to a bit of this register clears the corresponding bit in IPCx_ISR, and clears the interrupt caused by this bit. 0: Writing a '0' has no effect.
IPC0_CPUID	0x0010	RO	KM0	CPUID. Just IPC0 has this register. <ul style="list-style-type: none"> 0: KM0 has got the lock. 1: KM4 has got the lock.
IPCx_ISR	0x0014	RO	KM0/KM4	Interrupt Status Register It shows the current pending IPC interrupts.
IPC0_SEM	0x0018	R/W	KM0	Hardware Semaphore Register Just IPC0 has this register, for IPC1 this register is reserved, because KM0 still gets lock when KM4 sleeps.
IPCx_IER_R	0x001C	RO	KM0/KM4	Current IER Read Register The value of this register is the combination result of IPCx_IER & IPCx_IDR.
IPCx_USR	0x0020 ~ 0x004C	R/W	KM0/KM4	User-defined Register 12 dummy registers have been reserved for user. These registers can be used for H2L/L2H command defined by user.

7.3.1 IPCx_IER

- **Name:** Interrupt Enable Register
- **Size:** 32 bits
- **Address offset:** 0x0000
- **Read/write access:** write-only

The IPCx_IER (x = {0, 1}) is used to enable IPC interrupts, and the interrupt mask can be read from IPCx_IER_R.

31	30	29	28	27	...	4	3	2	1	0
IPC_IERx										
WO										

Bit	Name	Access	Default	Description
31:0	IPC_IERx	WO	0	<ul style="list-style-type: none"> Writing a '1' to a bit of this register enables the corresponding interrupt. Writing a '0' has no effect.

7.3.2 IPCx_IDR

- **Name:** Interrupt Disable Register
- **Size:** 32 bits
- **Address offset:** 0x0004
- **Read/write access:** write-only

The IPCx_IDR ($x = \{0, 1\}$) is used to disable IPC interrupts, and the interrupt mask can be read from IPCx_IER_R.

31	30	29	28	27	...	4	3	2	1	0
IPC_IDRx										
WO										

Bit	Name	Access	Default	Description
31:0	IPC_IDRx	WO	0	<ul style="list-style-type: none"> ● Writing a '1' to a bit of this register disables the corresponding interrupt. ● Writing a '0' has no effect.

7.3.3 IPCx_IRR

- **Name:** Interrupt Request Register
- **Size:** 32 bits
- **Address offset:** 0x0008
- **Read/write access:** write-only

The IPCx_IRR ($x = \{0, 1\}$) register allows other CPUs to send interrupt requests to the peer CPU. This is intended to allow communication between CPUs. For example, one CPU can be handling certain peripherals and signaling another CPU when data is available. The use of this feature is entirely up to the user.

31	30	29	28	27	...	4	3	2	1	0
IPC_IRRx										
WO										

Bit	Name	Access	Default	Description
31:0	IPC_IRRx	WO	0	<ul style="list-style-type: none"> ● Writing a '1' to a bit of this register requests the corresponding interrupt. ● Writing a '0' has no effect.

7.3.4 IPCx_ICR

- **Name:** Interrupt Clear Register
- **Size:** 32 bits
- **Address offset:** 0x000C
- **Read/write access:** write-only

The IPCx_ISR ($x = \{0, 1\}$) is used to clear pending IPC interrupts. It is set by the target CPU.

31	30	29	28	27	...	4	3	2	1	0
IPC_ICRx										
WO										

Bit	Name	Access	Default	Description
31:0	IPC_ICRx	WO	0	<ul style="list-style-type: none"> ● Writing a '1' to a bit of this register clears the corresponding interrupt. ● Writing a '0' has no effect.

7.3.5 IPC0_CPUID

- **Name:** CPU ID Register
- **Size:** 32 bits
- **Address offset:** 0x0010
- **Read/write access:** read-only

The IPC0_CPUID register can be used to check that the current CPU is KM0 or KM4. Just for IPC0 it is valid.

31	30	29	28	27	...	5	4	3	2	1	0
RSVD											IPC0_CPUID
											RO

Bit	Name	Access	Default	Description
31:1	RSVD	N/A	0	Reserved
0	IPC0_CPUID	RO	0	CPUID <ul style="list-style-type: none"> ● 0: KM0 is the current CPU. ● 1: KM4 is the current CPU.

7.3.6 IPCx_ISR

- **Name:** Current Interrupt Identifier Register
- **Size:** 32 bits
- **Address offset:** 0x0014
- **Read/write access:** read-only

The IPCx_ISR (x = {0, 1}) shows the current pending IPC interrupts.

31	30	29	28	27	...	4	3	2	1	0
IPC_ISRx										
RO										

Bit	Name	Access	Default	Description
31:0	IPC_ISRx	RO	0	<ul style="list-style-type: none"> ● 0: The corresponding interrupt source is not asserting the interrupt output currently. ● 1: The corresponding interrupt source is asserting the interrupt output currently.

7.3.7 IPC0_SEM

- **Name:** Hardware Semaphore Register
- **Size:** 32 bits
- **Address offset:** 0x0018
- **Read/write access:** read/write

The IPC0_SEM is a hardware semaphore control register, for IPC1 this register is reserved.

This register provides an Inter Processor Communication handshake. This can be used as a resource allocation handshake between 2 CPUs.

31	30	29	...	4	3	2	1	0
IPC_SEM_FREEx/IPC_SEM_CPUIDx (x = {1, 2, 3, ..., D, E, F})							IPC_SEM_CPUID0	IPC_SEM_FREE0
R/W							R/W	R/W

Bit	Name	Access	Default	Description
31:2	IPC_SEM_CPUIDF	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
30	IPC_SEM_FREEF	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
29	IPC_SEM_CPUIDE	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.

28	IPC_SEM_FREEE	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
27	IPC_SEM_CPUIDD	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
26	IPC_SEM_FREED	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
25	IPC_SEM_CPUIDC	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
24	IPC_SEM_FREEC	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
23	IPC_SEM_CPUIDB	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
22	IPC_SEM_FREEB	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
21	IPC_SEM_CPUIDA	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
20	IPC_SEM_FREEA	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
19	IPC_SEM_CPUID9	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
18	IPC_SEM_FREE9	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
17	IPC_SEM_CPUID8	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
16	IPC_SEM_FREE8	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
15	IPC_SEM_CPUID7	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
14	IPC_SEM_FREE7	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
13	IPC_SEM_CPUID6	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
12	IPC_SEM_FREE6	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
11	IPC_SEM_CPUID5	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
10	IPC_SEM_FREE5	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
9	IPC_SEM_CPUID4	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
8	IPC_SEM_FREE4	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
7	IPC_SEM_CPUID3	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
6	IPC_SEM_FREE3	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
5	IPC_SEM_CPUID2	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
4	IPC_SEM_FREE2	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
3	IPC_SEM_CPUID1	R/W	0	Refer to the description of the IPC_SEM_CPUID0 bit.
2	IPC_SEM_FREE1	R/W	0	Refer to the description of the IPC_SEM_FREE0 bit.
1	IPC_SEM_CPUID0	R/W	0	Indicates who owns the semaphore. ● 0: KM0 ● 1: KM4
0	IPC_SEM_FREE0	R/W	0	Indicates the status of the semaphore. Writing a '0' has no effect. ● 0: Free. When writing a '1', hardware sets the CPU ID. ● 1: Occupied. When writing a '1', hardware clears the 'CPUID' bit and the 'Free' bit.

7.3.8 IPCx_IER_R

- **Name:** Current IER Read Register
- **Size:** 32 bits
- **Address offset:** 0x001C
- **Read/write access:** read-only

The IPCx_IER_R (x = {0, 1}) is used to read current IPC interrupt mask. The value of this register is the combination result of IPCx_IER & IPCx_IDR.

31	30	29	28	27	...	4	3	2	1	0
IPC_IERx_R										
RO										

Bit	Name	Access	Default	Description
31:0	IPC_IERx_R	RO	0	<ul style="list-style-type: none"> ● 1: Interrupt is enabled for this bit. ● 0: Interrupt is disabled for this bit.

7.3.9 IPC_USR

- **Name:** User-defined Registers
- **Size:** 32 bits
- **Address offset:** 0x0020 ~ 0x004C
- **Read/write access:** read/write

These are user-defined register, and can be used for H2L/L2H command defined by user.

31	30	29	28	27	...	4	3	2	1	0
User-defined										
R/W										

Bit	Name	Access	Default	Description
31:0	User-defined	R/W	0	These registers can be used for H2L/L2H command defined by user.

8 General Purpose Input/Output (GPIO)

8.1 Introduction

GPIO is a programmable General Purpose Programming I/O peripheral.

Ameba-D GPIO IP controls the output data and direction of external I/O pads. It also can read back the data on external pads using memory-mapped registers. And Ameba-D supports two independent GPIO IP: PORT A (0~31) and PORT B (0~31).

Port A/B can be programmed to accept external signals as interrupt sources on any of the bits of the signal. The type of interrupt is programmable with one of the following settings:

8.1.1 General Product Description

Fig 8-1 shows the following functional groupings of the main interfaces to the GPIO block:

- APB interface to or from APB bridge
- External data Interface to or from I/O pads
- Auxiliary hardware data interface to or from auxiliary data sink or source
- Interrupt interface to or from interrupt controller

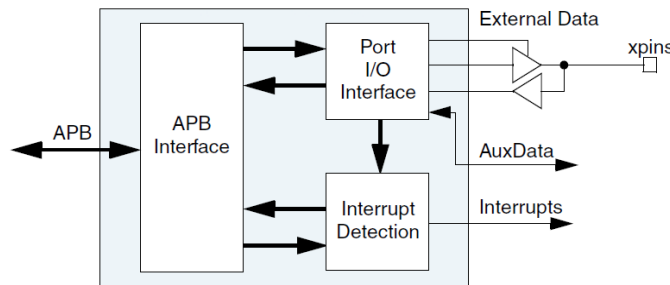


Fig 8-1 Block diagram

8.1.2 Features

GPIO supports the following features:

- Up to 32 independently configurable signals each port
- Separate data registers and data direction registers for each signal
- Configurable hardware and software control for each signal, or for each bit of each signal
- Separate auxiliary data input, data output, and data control for each I/O in Hardware Control mode
- Independently controllable signal bits
- Configurable interrupt mode for Port A/B
- Configurable debounce logic with an external slow clock to debounce interrupts
- Option to generate single or multiple interrupts
- GPIO Component Type register
- GPIO Component Version register
- Configurable reset values on output signals
- Configurable synchronization of interrupt signals

8.2 Functional Description

This chapter describes the functional operation of the GPIO.

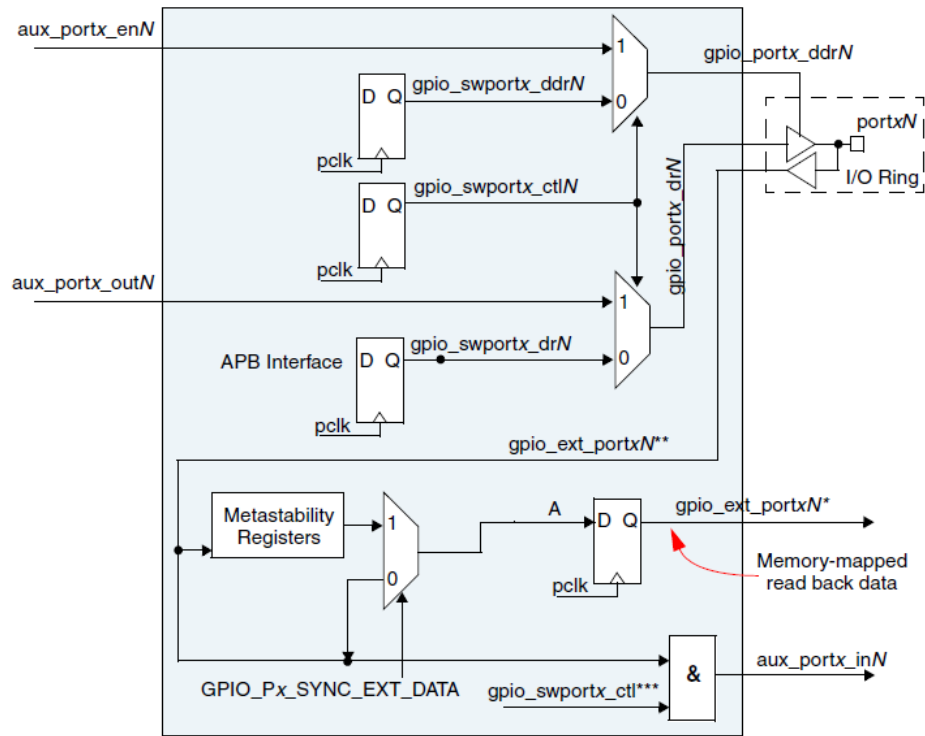
8.2.1 Data and Control Flow

GPIO controls the output data and direction of external I/O pads. It also can read back the data on external pads using memory-mapped registers.

There are two methods for generating the default source of the input data, the output data, and the control of each signal—through software or hardware control. Software control occurs over the APB bus interface; hardware control is through the auxiliary hardware control interface.

The software option always exists and is described in more detail in “Software Control Mode”. The hardware option for each signal exists only if you choose it during configuration time. Provided the hardware option is built, you can switch between software and hardware modes by writing to a control register for the corresponding signal. Also, the device can be configured so that you can individually switch between hardware and software modes for each bit of each signal, provided that hardware mode exists; for more detail, refer to “Hardware Control Mode”. The data and control flow for a signal are shown in Fig 8-2.

Note: Each bit in each signal is individually controllable. Therefore, each register can be thought of as N individual registers, where N is the signal width.



$N = 0$ through GPIO_PWIDTH_X where “X” is A, B, C or D.

* These data are multiplexed onto prdata when a read from the gpio_ext_portx register occurs.

** This is a port signal.

*** If configuration parameter $\text{GPIO_PORTN_SINGLE_CTL} = 0$, this remains a single bit as shown. If $\text{GPIO_PORTN_SINGLE_CTL} = 1$, this becomes a bus, gpio_swportx_ctlN .

Fig 8-2 Control RTL block diagram

Synchronization of the data to the bus clock is described in “Synchronization of Data to System Clock”. The default direction of any signal is set at configuration time. Later sections describe the I/O signals and memory-mapped registers in more detail.

8.2.1.1 Software Control Mode

When a signal is configured for software control, the data and direction control for the signal are sourced from the data register (gpio_swportx_dr) and direction control register (gpio_swportx_ddd), where x is a.

Under software control, the direction of the external I/O pad is controlled by a write to the Portx data direction register (gpio_swportx_ddr). The data written to this memory-mapped register gets mapped onto an output signal, gpio_portx_ddr, of the gpio peripheral. This output signal controls the direction of an external I/O pad.

The data written to the Portx data register (gpio_swportx_dr) drives the output buffer of the I/O pad. External data are input on the external data signal, gpio_ext_portx. Depending on whether gpio_ext_portx is configured as an input or output, the following occurs:

- Input – Reads the values on the signal
- Output – Reads the data register for Portx

The gpio_ext_portx register is read-only, meaning that it cannot be written from the APB software interface.

8.2.1.2 Hardware Control Mode

If a signal is configured for hardware control, it has external, auxiliary hardware signals controlling the data and the direction of Ports A through D. In this mode, the auxiliary data input signal (aux_portx_out) and direction control signal (aux_portx_en) are selected, where x is a.

The data direction of the external I/O pad, gpio_portx_ddr, is controlled through the auxiliary signal direction control signal, aux_portx_en.

For lines that are set to Output, the gpio_portx_dr and gpio_portx_ddr output signals drive the data and direction control onto the bidirectional pad that exists within the I/O ring of the SoC device. Fig 8-2 shows how the gpio peripheral controls the data and direction signals of an I/O PAD and data generation for the auxiliary source.

The gpio_swportx_ctl signal masks the value on the gpio_ext_portx external signal in order to generate aux_portx_in. The net result is that when hardware mode is selected, the value on aux_port_in output signal is equal to the value on the gpio_ext_portx input signal. When software mode is selected, the aux_portx_in output signal is always driven low. Setting bit 0 of gpio_swportx_ctl to 1 selects hardware mode for the entire signal if the parameter GPIO_PORTX_SINGLE_CTL is 1. If GPIO_PORTX_SINGLE_CTL is 0, setting bit n of gpio_swportx_ctl to 1 selects hardware mode for bit n of Portx. Setting bit 0 of gpio_swportx_ctl to 0 selects software mode for the entire signal if GPIO_PORTX_SINGLE_CTL is 1, while setting bit n of gpio_swportx_ctl to 0 selects software mode for bit n of Portx if GPIO_PORTX_SINGLE_CTL is 0. gpio_swportx_ctl to 0 selects software mode for bit n of Portx if GPIO_PORTX_SINGLE_CTL is 0.

8.2.1.3 Reading External Signals

The data on the external GPIO signal is read by an APB read of the memory-mapped register, gpio_ext_portx. An APB read to the pio_ext_portx register provides either the data on the gpio_ext_portx control lines or the contents of the gpio_swportx_dr, depending on the value of gpio_swportx_ddr.

Fig 8-2 shows how the hardware/software option is multiplexed, where the control lines for the multiplexing come from a memory-mapped register. It also shows the synchronization registers and the individual bit control of each data and data direction bit.

Note: The synchronization registers are optimized at the synthesis stage if the GPIO_PA_SYNC_EXT_DATA configuration parameter is equal to 0.

Fig 8-3 shows a timing diagram of a read to the gpio_ext_portx memory map registers when the direction is set to *Input* and the metastability registers are included.

Note: The maximum data rate that can be read back on consecutive reads from gpio_ext_portx must be less than pclk/2. Two pclk cycles per read access is required due to the non-pipelined nature of the APB. The assumption is that the APB bridge does not lose ownership of the AHB during consecutive accesses when PCLK=HCLK.

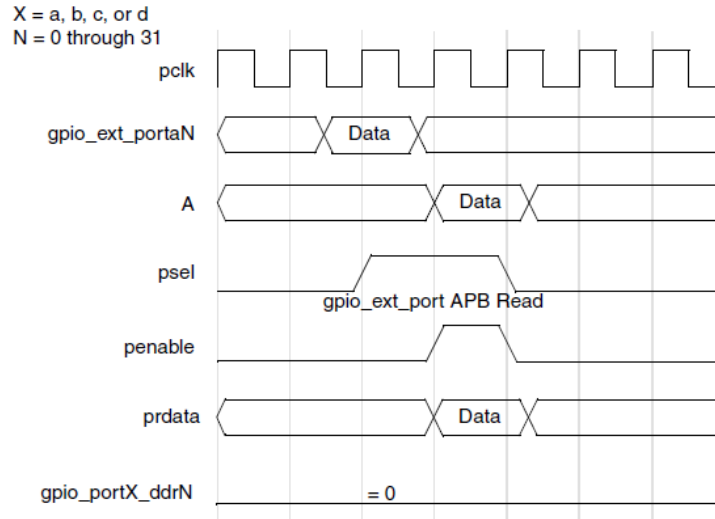


Fig 8-3 Read back of external `gpio_ext_portX` data timing

8.2.1.4 Synchronization of Data to System Clock

Synchronization of `gpio_ext_portX` to `pclk` prior to an APB read is enabled if the corresponding signal configuration parameter `GPIO_Px_SYNC_EXT_DATA` is set ($x = A, B, C, \text{ or } D$); this is shown in Fig 8-2.

8.2.2 Interrupts

Port A can be programmed to accept external signals as interrupt sources on any of the bits of the signal. The type of interrupt is programmable with one of the following settings:

- Active-high and level
- Active-low and level
- Rising edge
- Falling edge
- Both edge

The interrupts can be masked by programming the `gpio_intmask` register. The interrupt status can be read before masking (called raw status) and after masking.

The interrupts are also combined into a single interrupt output signal, which has the same polarity as the individual interrupts. Either individual interrupts (`gpio_intr` or `gpio_intr_n`) or a single combined interrupt (`gpio_intr_flag` or `gpio_intr_flag_n`) can be generated. In order to mask the combined interrupt, all individual interrupts have to be masked. The single combined interrupt does not have its own mask bit.

Whenever Port A is configured for interrupts, the data direction must be set to Input and the mode must be set to Software for interrupts to be latched. If the data direction register is reprogrammed to Output or the mode register is programmed to enable Hardware mode, then any pending interrupts are not lost. However, no new interrupts are generated.

Fig 8-4 illustrates how the interrupts are generated and how the data flows. The signal names in the diagram correspond to either I/O signals or memory-mapped registers.

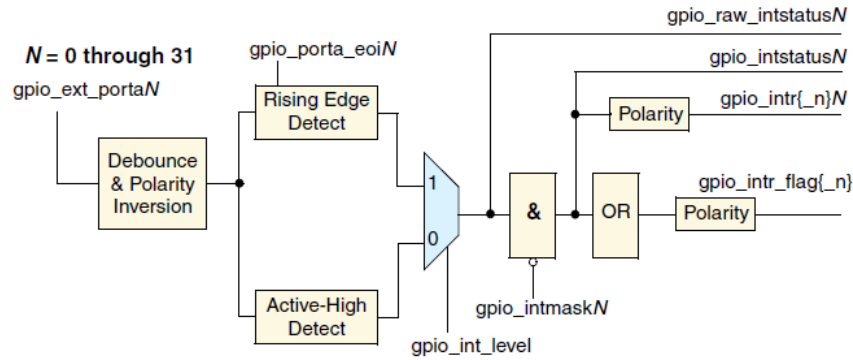


Fig 8-4 Interrupt RTL block diagram

Two interrupt request connection schemes are supported, and one scheme is chosen during configuration. The simplest connection scheme is where the combined interrupt `gpio_intr_flag` is generated by ORing together the bits of the `gpio_intr` bus. When only the combined interrupt request is used, then the `gpio_status` register must be read in the interrupt service routine (ISR) to find the source of the interrupt. When the individual interrupts lines are connected directly to the interrupt controller, then the `gpio_status` register does not have to be read by the ISR.

For edge-detected interrupts, the ISR can clear the interrupt by writing a 1 to the `gpio_porta_eoi` register for the corresponding bit to disable the interrupt. This write also clears the interrupt status and raw status registers. It is recommended that the interrupt source be cleared prior to writing to the `gpio_porta_eoi` register. Writing to the `gpio_porta_eoi` register has no effect on level-sensitive interrupts.

If level-sensitive interrupts cause the processor to interrupt, then the ISR can poll the `gpio_rawint` status register until the interrupt source disappears, or it can write to the `gpio_intmask` register to mask the interrupt before exiting the ISR. If the ISR exits without masking or disabling the interrupt prior to exiting, then the level-sensitive interrupt repeatedly requests an interrupt until the interrupt is cleared at the source.

If the `gpio_intr_flag` connection scheme is used and the interrupt service routine reads the `gpio_intr_status` register to find multiple pending interrupt requests, then it is up to the processor to prioritize these pending interrupt requests. There are no restrictions on the number of edge-detected interrupts that can be cleared simultaneously by writing multiple 1's to the `gpio_porta_eoi` register.

8.2.2.1 Debounce Operation

If the user has configured Port A to include the interrupt feature, GPIO can be configured to either include or exclude a debounce capability using the `GPIO_DEBOUNCE` parameter.

The external signal can be debounced to remove any spurious glitches that are less than one period of the external debouncing clock.

Fig 8-5 shows an RTL diagram of the debounce circuitry. The timing diagram shows an active-high input signal on `gpio_ext_portaN`. The polarity of the input signal detection is controlled by the memory-mapped signal, `gpio_int_polarity`. For a falling-edge or active-low-sensitive input, the input is then inverted and the same debounce logic is used as for rising-edge or active-high level-sensitive interrupts.

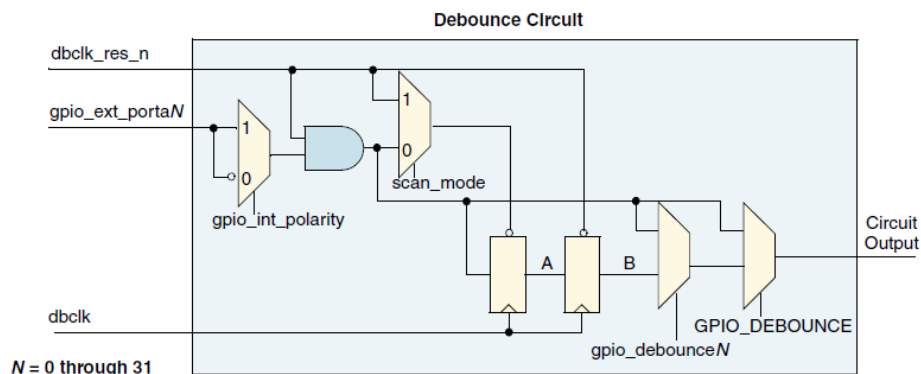


Fig 8-5 Debounce RTL diagram

A timing diagram of the debounce circuitry is shown in Fig 8-6.

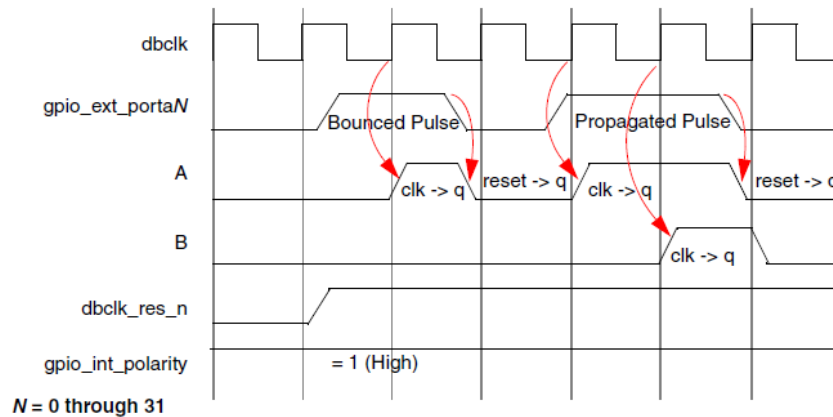


Fig 8-6 Debounce timing with asynchronous reset Flip-Flops

When input interrupt signals are debounced using a debounce clock, the signals must be active for a minimum of two cycles of the debounce clock to guarantee that they are registered. Any input pulse widths less than a debounce clock period are bounced. A pulse width between one and two debounce clock widths may or may not propagate, depending on its phase relationship to the debounce clock. If the input pulse spans two rising edges of the debounce clock, it is registered. If it spans only one rising edge, it is not registered.

The timing diagram in Fig 8-6 shows both cases: the input signal being bounced, and later, a propagated input signal. If GPIO supports debounce, then debouncing input signals on Port A can be enabled or disabled under software control.

The **dbclk_res_n** signal is asynchronously asserted and synchronously de-asserted to the debounce clock, **dbclk**. The system reset signal, **presn**, is asynchronously asserted and synchronously de-asserted to **pclk**; synchronization must be external to the component. The **pclk** and **dbclk** signals are assumed to be asynchronous to each other.

Note: The use of the debounce circuitry increases interrupt latency by two clock cycles of the debounce clock.

The debounce circuitry works with only asynchronous reset flip-flops.

8.2.2.2 Synchronization of Interrupt Signals to System Clock

Interrupt signals can be internally synchronized to a system clock, **pclk_intr**. Synchronization to **pclk_intr** must occur for edge-detect signals. Edge-detected interrupts to the processor are always synchronous to the system bus clock. With level-sensitive interrupts, synchronization is optional and under software control.

The **pclk_intr** signal is needed for systems that may have the GPIO **pclk** bus clock gated off, but the system still wants to detect interrupts. It is assumed that this clock is synchronous to **pclk**. If interrupt detection is required only when **pclk** is running, then **pclk_intr** and **pclk** can be connected to the same clock source. If the system employs a gated **pclk** to the gpio, **pclk_intr** needs to be running for interrupt detection to occur.

The **gpio_intrclk_en** output signal is asserted when either edge-sensitive interrupts or level-sensitive interrupts requiring synchronization are enabled in the GPIO block. Both cases require a clock for detection. Therefore, this signal can cause the external clock generator block to generate **pclk_intr**.

8.2.2.3 Interrupt Edge Detection

8.2.2.3.1 Single Edge

Fig 8-7 shows an RTL diagram of the synchronization and edge detection of interrupt sources on **gpio_ext_portaN** signals, when **GPIO_INT_BOTH_EDGE=0**.

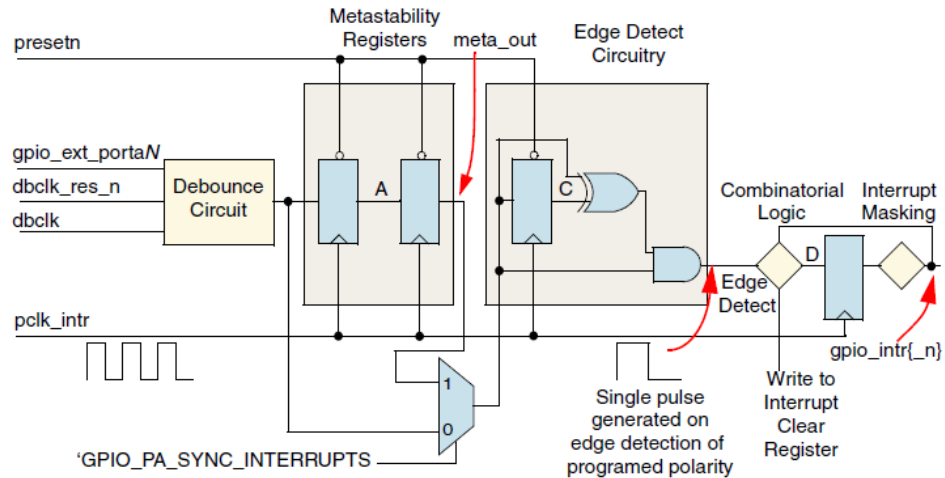


Fig 8-7 Synchronization and edge detect interrupt generation when GPIO_INT_BOTH_EDGE=0

The MUX allows inclusion or exclusion of the metastability registers at configuration depending on the value of GPIO_PA_SYNC_INTERRUPTS. If this parameter is a 1, the registers are included.

Fig 8-8 shows a timing diagram in which an interrupt is generated on the rising edge of an input on Port A; this is where the debounce logic is disabled and metastability registers are included. It also shows how an interrupt is cleared by a write to the interrupt clear register.

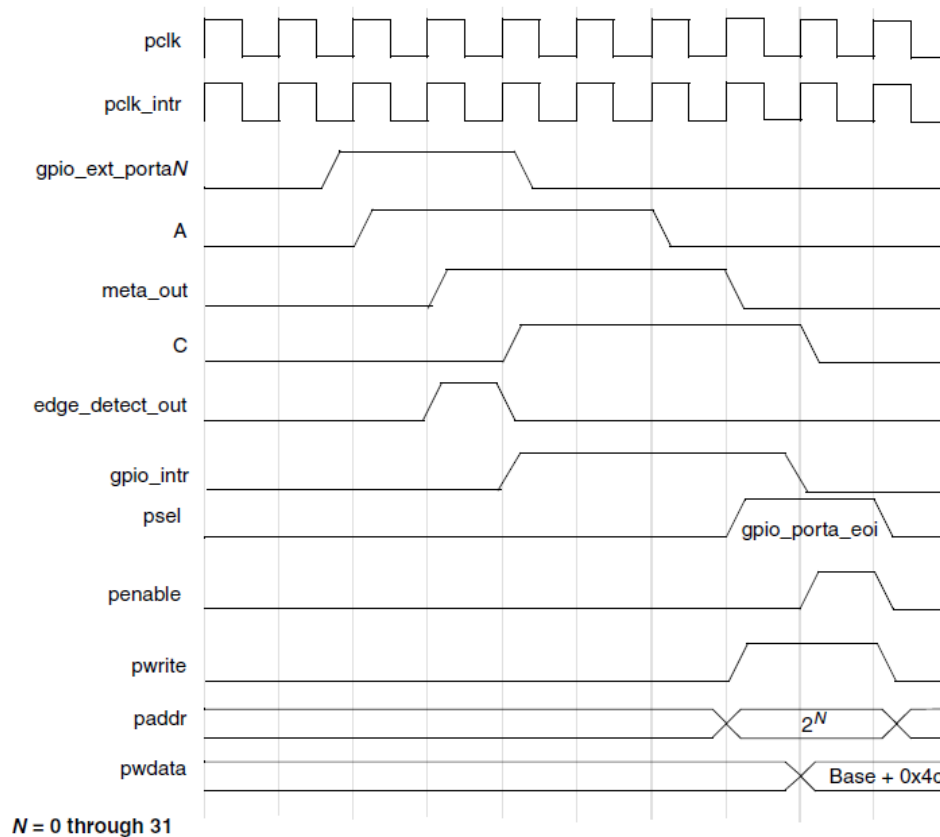


Fig 8-8 Interrupt edge detection and interrupt clear timing when GPIO_SYNC_PA_INTERRUPTS = 1 (metastability included)

A case may arise where the Interrupt Service Routine (ISR) writes to the interrupt clear register in order to clear an existing interrupt during the same clock cycle in which a new interrupt is detected. In such a case, writing to the interrupt clear register clears only the first interrupt. The second interrupt is not lost, since setting an interrupt has a higher priority than clearing it.

Fig 8-9 shows a timing diagram similar to Fig 8-8, except that metastability registers are removed. It also shows how an interrupt is cleared by a write to the interrupt clear register.

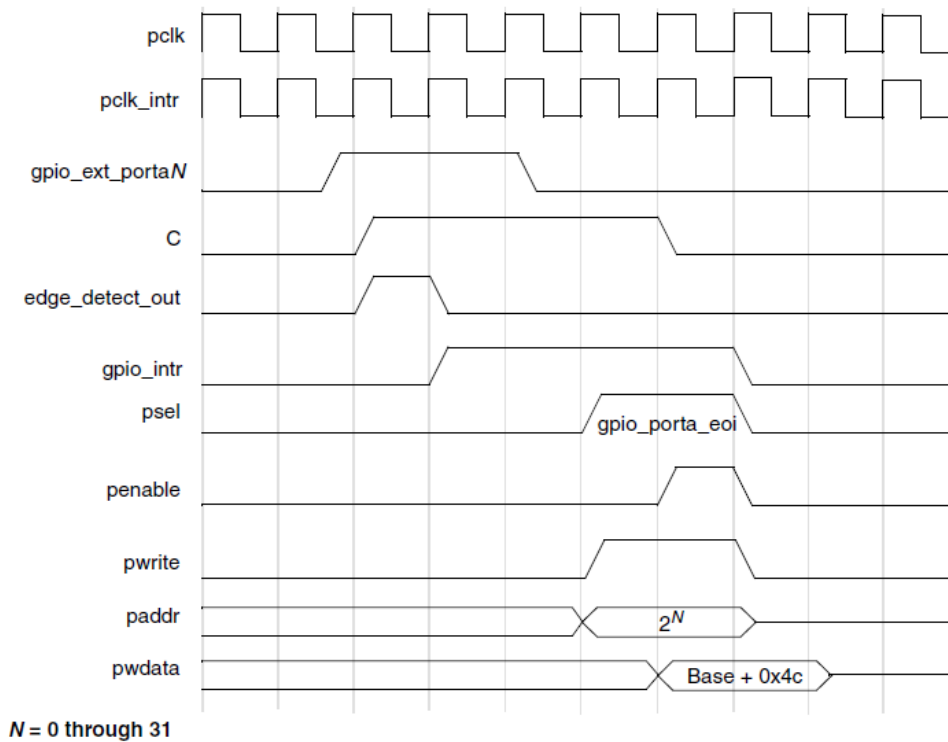


Fig 8-9 Interrupt edge detection and interrupt clear timing when GPIO_SYNC_PA_INTERRUPTS = 0 (metastability removed)

Note: Since the Metastability registers are removed from the path, A and B waveforms shown in Fig 8-8 no longer exist, and all subsequent signal delays are reduced by two clock cycles.

Fig 8-10 shows such a case where the debounce logic is unused. In this timing diagram, meta_out and edge_detect_out are the outputs of the second metastability register and the edge detect logic, respectively. The second edge detection occurs on the same cycle as the write to the interrupt clear register. In this example, the write to the interrupt clear register does not clear the second interrupt, and the gpio_intr{_n} signal is not de-asserted.

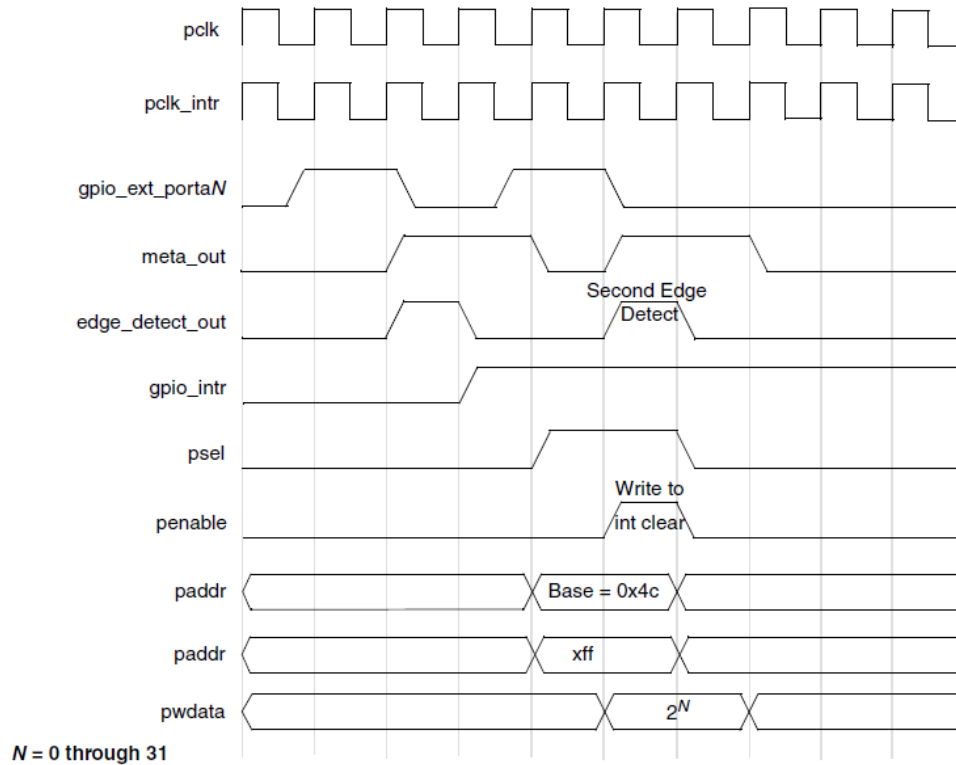


Fig 8-10 Write to interrupt clear register, coincident with detection of new interrupt

8.2.2.3.2 Both Edges

Interrupt detection logic for both the rising edge and the falling edge is available only when GPIO_INT_BOTH_EDGE = 1 and this logic detects the interrupt on both the rising edge and the falling edge when the gpio_int_bothedge register is programmed to 1.

Fig 8-11 shows the synchronization and edge detect interrupt generation of the interrupt sources on gpio_ext_portaN signals when GPIO_INT_BOTH_EDGE = 1.

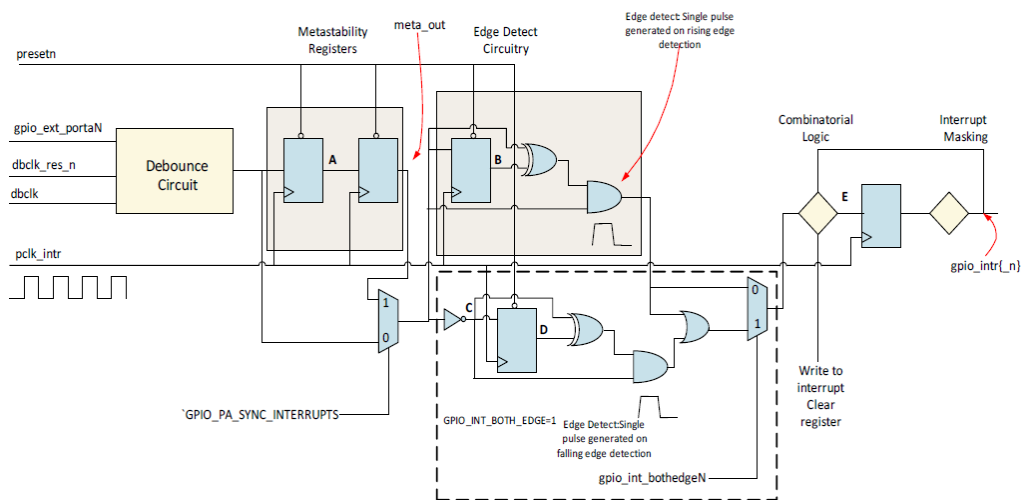


Fig 8-11 Synchronization and edge detect interrupt generation when GPIO_INT_BOTH_EDGE=0

Fig 8-12 shows a timing diagram where an interrupt is generated on both the rising edge and the falling edge of an input on Port A, that is, with `GPIO_INT_BOTH_EDGE = 1` and `gpio_int_bothedge` programmed to detect both edges. In this scenario, debounce logic is disabled and metastability registers are included. This figure also shows how an interrupt is cleared by a write to the interrupt clear register.

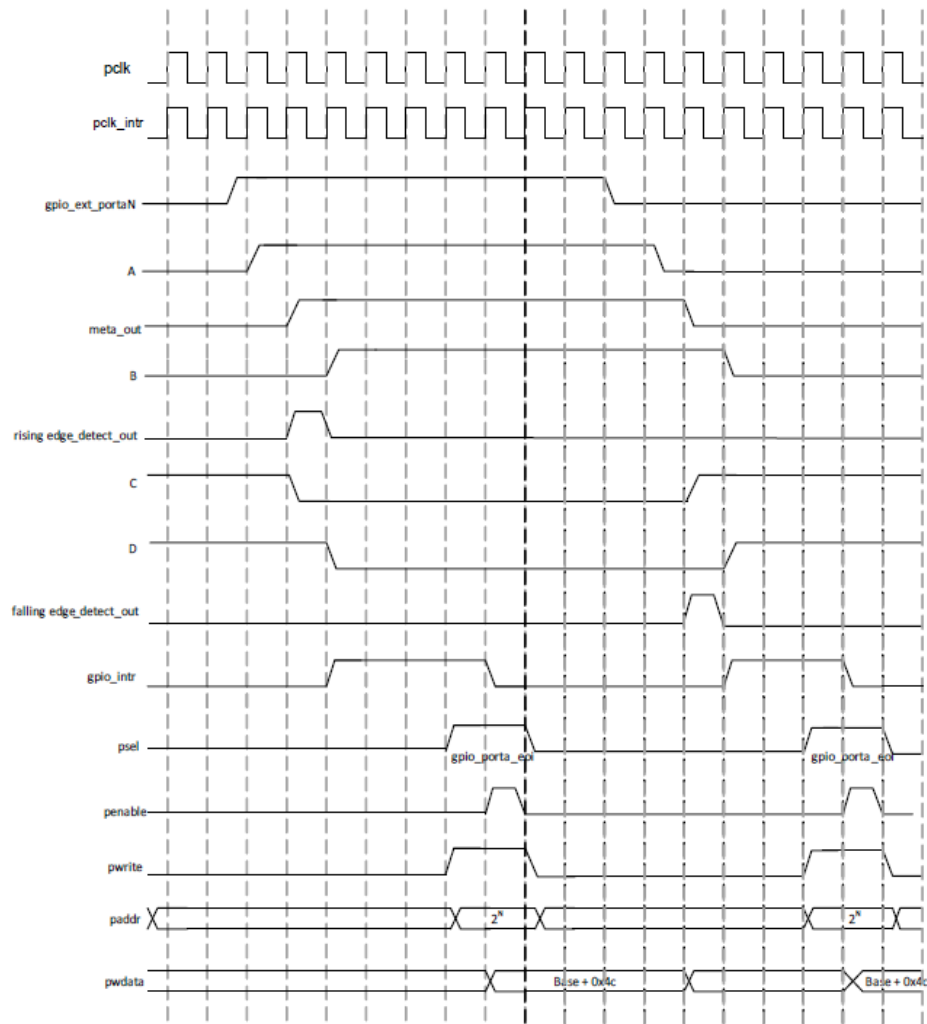


Fig 8-12 Interrupt edge detection and interrupt clear timing when `GPIO_SYNC_PA_INTERRUPTS = 1` and `GPIO_INT_BOTH_EDGE=1` (metastability included)

Fig 8-13 shows a timing diagram similar to Fig 8-12, except that in this scenario, metastability registers are removed.

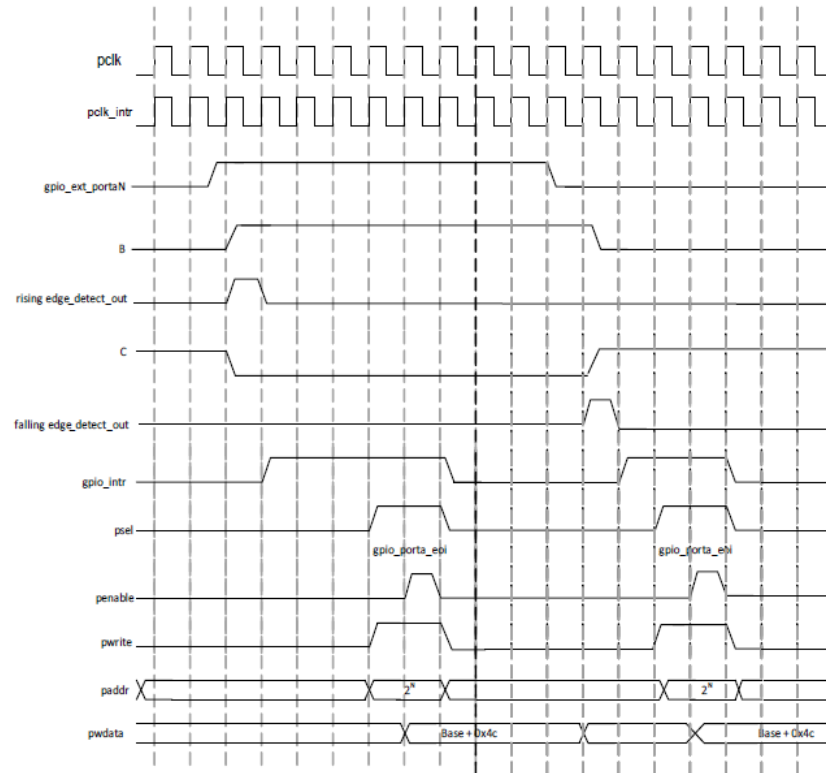


Fig 8-13 Interrupt edge detection and interrupt clear timing when GPIO_SYNC_PA_INTERRUPTS = 0 and GPIO_INT_BOTH_EDGE=1 (metastability Removed)

8.2.2.4 Level-Sensitive Interrupts

Fig 8-14 shows the generation of level-sensitive interrupts. As for edge-detect interrupts, the user can configure GPIO with or without debounce logic.

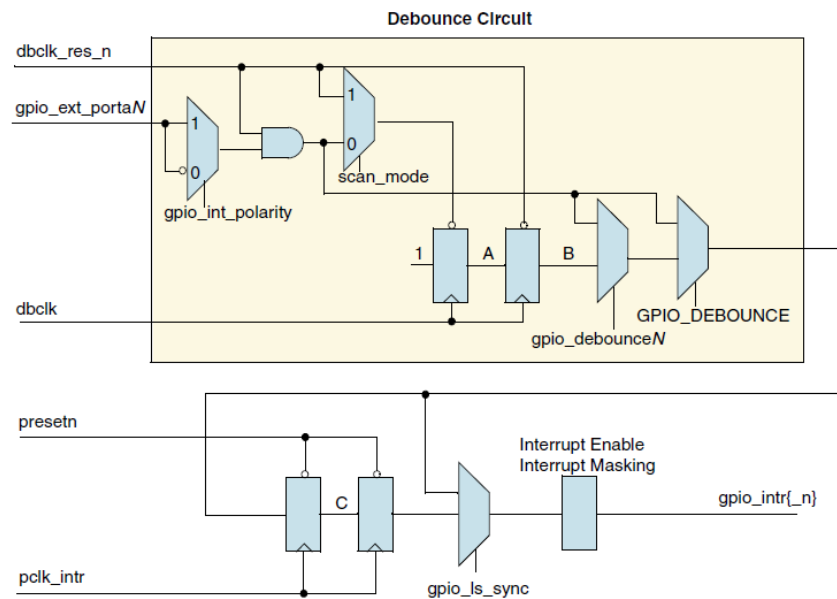


Fig 8-14 Level-sensitive interrupt RTL diagram

With level-sensitive interrupts, there is a choice of whether they are synchronized to the interrupt clock `pclk_intr` or are entirely combinational (aside from the debounce circuitry). The selection is done by programming the `gpio_ls_sync` (GPIO Level Sensitive Synchronous) register.

This is a memory-mapped bit that inserts two metastability registers clocked off of `pclk_intr` to synchronize the level-sensitive interrupts to `pclk_intr`. When `gpio_ls_sync` is not asserted, then there is no guarantee that the interrupt lines are synchronous to `pclk_intr`. A processor status register may need to be set to indicate asynchronous interrupts. When `gpio_ls_sync` is asserted, then the `pclk_intr` clock must be present to pass the interrupt to the interrupt controller block. The `gpio_intrclk_en` output signal is asserted when level-sensitive interrupts that are to be synchronized to `pclk_intr` are selected. The `gpio_intrclk_en` signal can be used in the clock generation block to turn on `pclk_intr`.

The input signal is inverted for active-low level-sensitive interrupts. The same detection logic is used for active-high level-sensitive interrupts.

Fig 8-15 shows the generation of an active-low level-sensitive interrupt where the debounce circuitry is disabled.

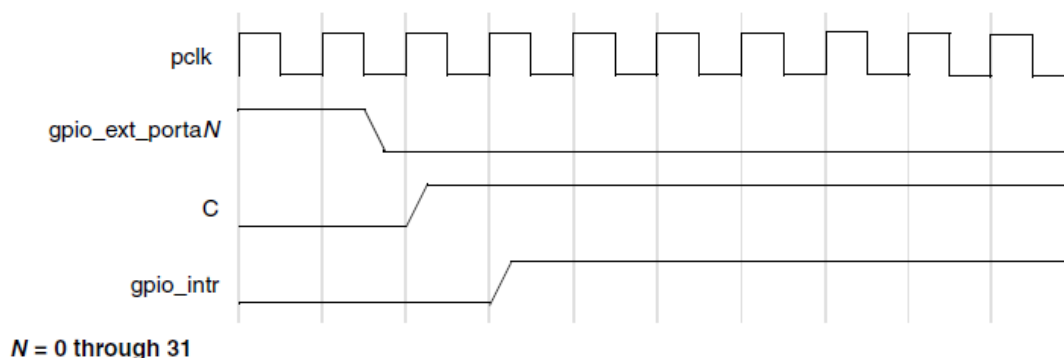


Fig 8-15 Active-low level-sensitive interrupt generation timing

8.3 Registers

This section describes the programmable registers of the GPIO.

8.3.1 Bus Interface

GPIO peripheral has a standard AMBA 2.0 APB interface for reading and writing the internal registers. This peripheral supports APB data bus widths of 8, 16, or 32 bits, which is set with the `APB_DATA_WIDTH` parameter.

Fig 8-16 shows the read/write busses between the APB and the APB slave.

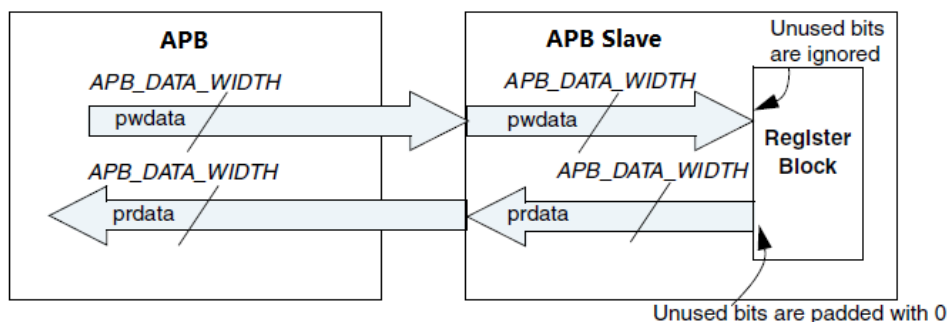


Fig 8-16 Relationship between APB and APB slave data widths

8.3.2 Register Memory Map

Table 8-1 shows the memory map for the GPIO peripheral.

Table 8-1 Memory map of GPIO

Name	Address Offset	Access	Description
gpio_swporta_dr	0x00	R/W	Port A data register Width: <i>GPIO_PWIDTH_A</i> Reset Value: <i>GPIO_SWPORTA_RESET</i>
gpio_swporta_ddr	0x04	R/W	Port A data direction register Width: <i>GPIO_PWIDTH_A</i> Reset Value: <i>GPIO_DFLT_DIR_A</i> (for all bits)
gpio_swporta_ctl	0x08	R/W	Port A data source register Width: 1 bit if <i>GPIO_PORTA_SINGLE_CTL</i> = 1, or <i>GPIO_PWIDTH_A</i> otherwise Reset Value: <i>GPIO_DFLT_SRC_A</i> Bit is repeated <i>GPIO_PWIDTH_A</i> times if <i>GPIO_PORTA_SINGLE_CTL</i> = 0
gpio_swportb_dr	0x0c	R/W	Port B data register Width: <i>GPIO_PWIDTH_B</i> Reset Value: <i>GPIO_SWPORTB_RESET</i>
gpio_swportb_ddr	0x10	R/W	Port B data direction register Width: <i>GPIO_PWIDTH_B</i> Reset Value: <i>GPIO_DFLT_DIR_B</i> (for all bits)
gpio_swportb_ctl	0x14	R/W	Port B data source register Width: 1 bit if <i>GPIO_PORTB_SINGLE_CTL</i> = 1, or <i>GPIO_PWIDTH_B</i> otherwise Reset Value: <i>GPIO_DFLT_SRC_B</i> Bit is repeated <i>GPIO_PWIDTH_B</i> times if <i>GPIO_PORTB_SINGLE_CTL</i> = 0
gpio_swportc_dr	0x18	R/W	Port C data register Width: <i>GPIO_PWIDTH_C</i> Reset Value: <i>GPIO_SWPORTC_RESET</i>
gpio_swportc_ddr	0x1c	R/W	Port C data direction register Width: <i>GPIO_PWIDTH_C</i> Reset Value: <i>GPIO_DFLT_DIR_C</i> (for all bits)
gpio_swportc_ctl	0x20	R/W	Port C data source register Width: 1 bit if <i>GPIO_PORTC_SINGLE_CTL</i> = 1, or <i>GPIO_PWIDTH_C</i> otherwise Reset Value: <i>GPIO_DFLT_SRC_C</i> Bit is repeated <i>GPIO_PWIDTH_C</i> times if <i>GPIO_PORTC_SINGLE_CTL</i> = 0
gpio_swportd_dr	0x24	R/W	Port D data register Width: <i>GPIO_PWIDTH_D</i> Reset Value: <i>GPIO_SWPORTD_RESET</i>
gpio_swportd_ddr	0x28	R/W	Port D data direction register Width: <i>GPIO_PWIDTH_D</i> Reset Value: <i>GPIO_DFLT_DIR_D</i> (for all bits)
gpio_swportd_ctl	0x2c	R/W	Port D data source register Width: 1 bit if <i>GPIO_PORTD_SINGLE_CTL</i> = 1, or <i>GPIO_PWIDTH_D</i> otherwise Reset Value: <i>GPIO_DFLT_SRC_D</i> Bit is repeated <i>GPIO_PWIDTH_D</i> times if <i>GPIO_PORTD_SINGLE_CTL</i> = 0
gpio_inten	0x30	R/W	Interrupt enable register Width: <i>GPIO_PWIDTH_A</i> Reset Value: 0x0
gpio_intmask	0x34	R/W	Interrupt mask register Width: <i>GPIO_PWIDTH_A</i> Reset Value: 0x0
gpio_inttype_level	0x38	R/W	Interrupt level register Width: <i>GPIO_PWIDTH_A</i> Reset Value: 0x0
gpio_int_polarity	0x3c	R/W	Interrupt polarity register

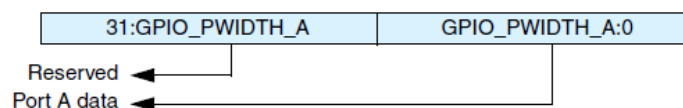
			Width: <i>GPIO_PWIDTH_A</i> Reset Value: 0x0
gpio_intstatus	0x40	R	Interrupt status of Port A Width: <i>GPIO_PWIDTH_A</i> Reset Value: 0x0
gpio_raw_intstatus	0x44	R	Raw interrupt status of Port A (premasking) Width: <i>GPIO_PWIDTH_A</i> Reset Value: 0x0
gpio_debounce	0x48	R/W	Debounce enable register Width: <i>GPIO_PWIDTH_A</i> Reset Value: 0x0
gpio_porta_eoi	0x4c	W	Port A clear interrupt register Width: <i>GPIO_PWIDTH_A</i> Reset Value: 0x0
gpio_ext_porta	0x50	R	Port A external port register Width: <i>GPIO_PWIDTH_A</i> Reset Value: 0x0
gpio_ext_portb	0x54	R	Port B external port register Width: <i>GPIO_PWIDTH_B</i> Reset Value: 0x0
gpio_ext_portc	0x58	R	Port C external port register Width: <i>GPIO_PWIDTH_C</i> Reset Value: 0x0
gpio_ext_portd	0x5c	R	Port D external port register Width: <i>GPIO_PWIDTH_D</i> Reset Value: 0x0
gpio_ls_sync	0x60	R/W	Level-sensitive synchronization enable register Reset Value: 0x0
gpio_id_code	0x64	R	ID code register Width: <i>GPIO_ID_WIDTH</i> Reset Value: <i>GPIO_ID_NUM</i>
Gpio_int_bothedge	0x68	R/W	Interrupt Both Edge register Width: <i>GPIO_PWIDTH_A</i> Reset Value: 0x0
gpio_ver_id_code	0x6c	R	Component Version register Reset Value: See the Releases table in the <i>Release Notes</i>
gpio_config_reg2	0x70	R	Configuration Register 2 Reset Value: Reset value depends on configuration parameters.
gpio_config_reg1	0x74	R	Configuration Register 1 Reset Value: Reset value depends on configuration parameters.

8.3.3 Register and Field Descriptions

The following sections contain the memory diagrams and field descriptions for the individual registers.

8.3.3.1 gpio_swporta_dr

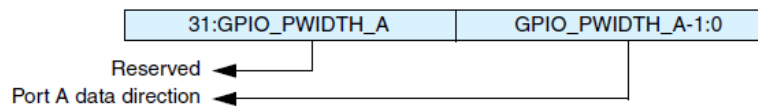
- **Name:** Port A Data Register
- **Size:** *GPIO_PWIDTH_A*
- **Address offset:** 0x00
- **Read/write access:** read/write



Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	Port A Data Register	R/W	Values written to this register are output on the I/O signals for Port A if the corresponding data direction bits for Port A are set to Output mode and the corresponding control bit for Port A is set to Software mode. The value read back is equal to the last value written to this register. Reset Value: GPIO_SWPORTA_RESET

8.3.3.2 gpio_swporta_ddr

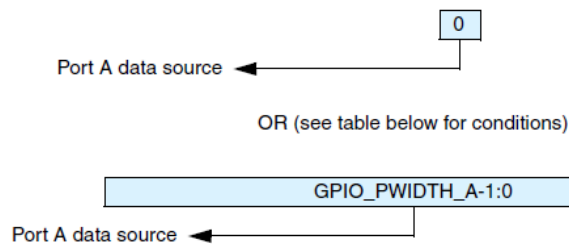
- **Name:** Port A Data Direction Register
- **Size:** GPIO_PWIDTH_A
- **Address offset:** 0x04
- **Read/write access:** read/write



Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	Port A Data Direction Register	R/W	Values written to this register independently control the direction of the corresponding data bit in Port A. The default direction can be configured as input or output after system reset through the GPIO_DFLT_DIR_A parameter. 0 – Input (default) 1 – Output Reset Value: GPIO_DFLT_DIR_A

8.3.3.3 gpio_swporta_ctl

- **Name:** Port A Data Source
- **Size:**
 - 1 bit wide if GPIO_PORTA_SINGLE_CTL = 1
 - GPIO_PWIDTH_A bits wide if GPIO_PORTA_SINGLE_CTL = 0
- **Address offset:** 0x08
- **Read/write access:** read/write

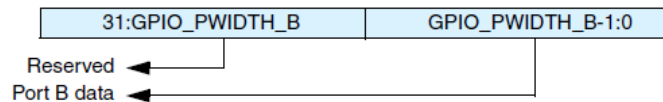


Bit	Name	Access	Description
0 -or- 0:GPIO_PWIDTH_A-1 (See above)	Port A Data Source	R/W	The data and control source for a signal can come from either software or hardware; this bit selects between them. The default source is configurable through the GPIO_DFLT_SRC_A configuration parameter. 0 – Software mode (default) 1 – Hardware mode

			<p>If GPIO_PORTA_SINGLE_CTL = 0, the register will contain one bit for each bit of the signal. Upon reset in this case, the value of GPIO_DFLT_SRC_A is replicated across all bits of the signal so that all bits power up with the same operating mode. Furthermore, the default source of each bit of the signal can subsequently be changed by writing to the corresponding bit of this register.</p> <p>This register is not available unless GPIO_HW_PORTA = 1.</p> <p>Reset Value: If GPIO_PORTA_SINGLE_CTL = 1, then the reset value is GPIO_DFLT_SRC_A.</p> <p>If GPIO_PORTA_SINGLE_CTL = 0, then the reset value is {GPIO_PWIDTH_A{GPIO_DFLT_SRC_A in each bit}}.</p>
--	--	--	---

8.3.3.4 gpio_swportb_dr

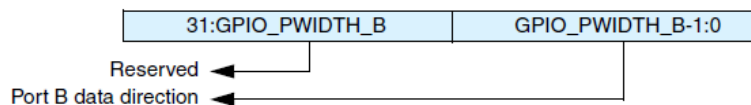
- **Name:** Port B Data Register
- **Size:** GPIO_PWIDTH_B
- **Address offset:** 0x0C
- **Read/write access:** read/write



Bit	Name	Access	Description
31:GPIO_PWIDTH_B	Reserved	Read as zero	Reserved
GPIO_PWIDTH_B-1:0	Port B Data Register	R/W	<p>Values written to this register are output on the I/O signals for Port B if the corresponding data direction bits for Port B are set to Output mode and the corresponding control bit for Port B is set to Software mode. The value read back is equal to the last value written to this register.</p> <p>Reset Value: GPIO_SWPORTB_RESET</p>

8.3.3.5 gpio_swportb_ddr

- **Name:** Port B Data Direction
- **Size:** GPIO_PWIDTH_B
- **Address offset:** 0x10
- **Read/write access:** read/write

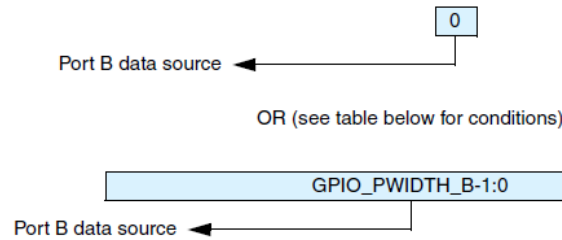


Bit	Name	Access	Description
31:GPIO_PWIDTH_B	Reserved	Read as zero	Reserved
GPIO_PWIDTH_B-1:0	Port B Data Direction Register	R/W	<p>Values written to this register independently control the direction of the corresponding data bit in Port B. The default direction can be configured as input or output after system reset through the GPIO_DFLT_DIR_B parameter.</p> <p>0 – Input (default) 1 – Output</p> <p>Reset Value: GPIO_DFLT_DIR_B</p>

8.3.3.6 gpio_swportb_ctl

- **Name:** Port B Data Source
- **Size:**

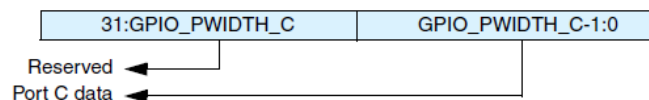
- 1 bit wide if GPIO_PORTB_SINGLE_CTL = 1
- GPIO_PWIDTH_B bits wide if GPIO_PORTB_SINGLE_CTL = 0
- **Address offset:** 0x14
- **Read/write access:** read/write



Bit	Name	Access	Description
0 -or- 0:GPIO_PWIDTH_B-1 (See above)	Port B Data Source	R/W	<p>The data and control source for a signal can come from either software or hardware; this bit selects between them. The default source is configurable through the GPIO_DFLT_SRC_B configuration parameter.</p> <p>0 – Software mode (default)</p> <p>1 – Hardware mode</p> <p>If GPIO_PORTB_SINGLE_CTL = 0, the register will contain one bit for each bit of the signal. Upon reset in this case, the value of GPIO_DFLT_SRC_B is replicated across all bits of the signal so that all bits power up with the same operating mode. Furthermore, the default source of each bit of the signal can subsequently be changed by writing to the corresponding bit of this register.</p> <p>This register is not available unless GPIO_HW_PORTB = 1.</p> <p>Reset Value: If GPIO_PORTB_SINGLE_CTL = 1, then the reset value is GPIO_DFLT_SRC_B.</p> <p>If GPIO_PORTB_SINGLE_CTL = 0, then the reset value is {GPIO_PWIDTH_B{GPIO_DFLT_SRC_B in each bit}}.</p>

8.3.3.7 gpio_swportc_dr

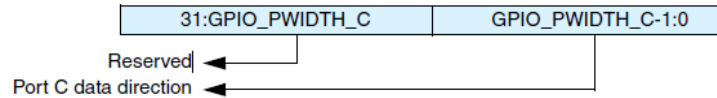
- **Name:** Port C Data Register
- **Size:** GPIO_PWIDTH_C
- **Address offset:** 0x18
- **Read/write access:** read/write



Bit	Name	Access	Description
31:GPIO_PWIDTH_C	Reserved	Read as zero	Reserved
GPIO_PWIDTH_C-1:0	Port C Data Register	R/W	<p>Values written to this register are output on the I/O signals for Port C if the corresponding data direction bits for Port C are set to Output mode and the corresponding control bit for Port C is set to Software mode. The value read back is equal to the last value written to this register.</p> <p>Reset Value: GPIO_SWPORTC_RESET</p>

8.3.3.8 gpio_swportc_ddr

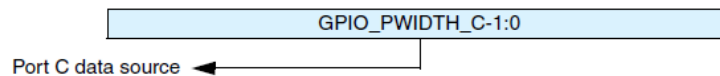
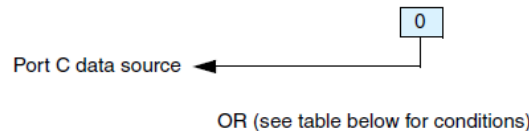
- **Name:** Port C Data Direction
- **Size:** GPIO_PWIDTH_C
- **Address offset:** 0x1C
- **Read/write access:** read/write



Bit	Name	Access	Description
31:GPIO_PWIDTH_C	Reserved	Read as zero	Reserved
GPIO_PWIDTH_C-1:0	Port C Data Direction	R/W	Values written to this register independently control the direction of the corresponding data bit in Port C. The default direction can be configured as input or output after system reset through the GPIO_DFLT_DIR_C parameter. 0 – Input (default) 1 – Output Reset Value: GPIO_DFLT_DIR_C

8.3.3.9 gpio_swportc_ctl

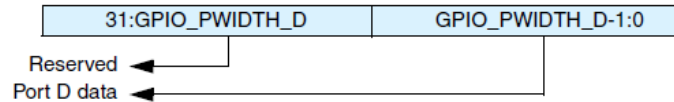
- **Name:** Port C Data Source
- **Size:**
 - 1 bit wide if GPIO_PORTC_SINGLE_CTL = 1
 - GPIO_PWIDTH_C bits wide if GPIO_PORTC_SINGLE_CTL = 0
- **Address offset:** 0x20
- **Read/write access:** read/write



Bit	Name	Access	Description
0 -or- 0:GPIO_PWIDTH_C-1 (See above)	Port C Data Source	R/W	The data and control source for a signal can come from either software or hardware; this bit selects between them. The default source is configurable through the GPIO_DFLT_SRC_C configuration parameter. 0 – Software mode (default) 1 – Hardware mode If GPIO_PORTC_SINGLE_CTL = 0, the register will contain one bit for each bit of the signal. Upon reset in this case, the value of GPIO_DFLT_SRC_C is replicated across all bits of the signal so that all bits power up with the same operating mode. Furthermore, the default source of each bit of the signal can subsequently be changed by writing to the corresponding bit of this register. This register is not available unless GPIO_HW_PORTC = 1. Reset Value: If GPIO_PORTC_SINGLE_CTL = 1, then the reset value is GPIO_DFLT_SRC_C. If GPIO_PORTC_SINGLE_CTL = 0, then the reset value is {GPIO_PWIDTH_C{GPIO_DFLT_SRC_C in each bit}}.

8.3.3.10 gpio_swportd_dr

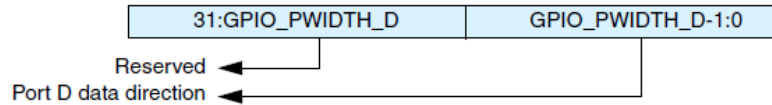
- **Name:** Port D Data Register
- **Size:** GPIO_PWIDTH_D
- **Address offset:** 0x24
- **Read/write access:** read/write



Bit	Name	Access	Description
<code>31:GPIO_PWIDTH_D</code>	Reserved	Read as zero	Reserved
<code>GPIO_PWIDTH_D-1:0</code>	Port D Data Register	R/W	Values written to this register are output on the I/O signals for Port D if the corresponding data direction bits for Port D are set to Output mode and the corresponding control bit for Port D is set to Software mode. The value read back is equal to the last value written to this register. Reset Value: <code>GPIO_SWPORTD_RESET</code>

8.3.3.11 gpio_swportd_dds

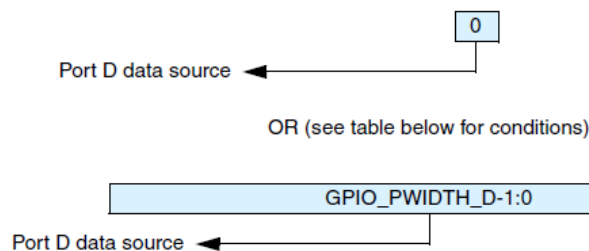
- **Name:** Port D Data Direction
- **Size:** `GPIO_PWIDTH_D`
- **Address offset:** 0x28
- **Read/write access:** read/write



Bit	Name	Access	Description
<code>31:GPIO_PWIDTH_D</code>	Reserved	Read as zero	Reserved
<code>GPIO_PWIDTH_D-1:0</code>	Port D Data Direction	R/W	Values written to this register independently control the direction of the corresponding data bit in Port D. The default direction can be configured as input or output after system reset through the <code>GPIO_DFLT_DIR_D</code> parameter. 0 – Input (default) 1 – Output Reset Value: <code>GPIO_DFLT_DIR_D</code>

8.3.3.12 gpio_swportd_ctl

- **Name:** Port D Data Source
- **Size:**
 - 1 bit wide if `GPIO_PORTD_SINGLE_CTL = 1`
 - `GPIO_PWIDTH_D` bits wide if `GPIO_PORTD_SINGLE_CTL = 0`
- **Address offset:** 0x2C
- **Read/write access:** read/write



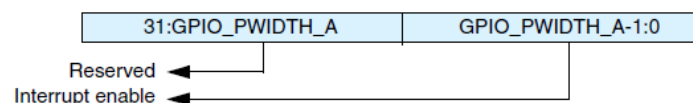
Bit	Name	Access	Description
-----	------	--------	-------------

0 -or- 0:GPIO_PWIDTH_D-1 (See above)	Port D Data Source	R/W	<p>The data and control source for a signal can come from either software or hardware; this bit selects between them. The default source is configurable through the GPIO_DFLT_SRC_D configuration parameter.</p> <p>0 – Software mode (default) 1 – Hardware mode</p> <p>If GPIO_PORTD_SINGLE_CTL = 0, the register will contain one bit for each bit of the signal. Upon reset in this case, the value of GPIO_DFLT_SRC_D is replicated across all bits of the signal so that all bits power up with the same operating mode. Furthermore, the default source of each bit of the signal can subsequently be changed by writing to the corresponding bit of this register.</p> <p>This register is not available unless GPIO_HW_PORTD = 1.</p> <p>Reset Value: If GPIO_PORTD_SINGLE_CTL = 1, then the reset value is <i>GPIO_DFLT_SRC_D</i>. If GPIO_PORTD_SINGLE_CTL = 0, then the reset value is {GPIO_PWIDTH_D{GPIO_DFLT_SRC_D in each bit}}.</p>
---	--------------------	-----	--

8.3.3.13 gpio_inten

- **Name:** Interrupt enable
- **Size:** GPIO_PWIDTH_A
- **Address offset:** 0x30
- **Read/write access:** read/write

This register is available only if Port A is configured to generate interrupts (GPIO_PORTA_INTR = Include (1)).

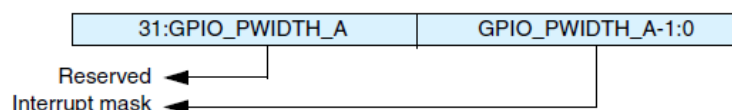


Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	Interrupt enable	R/W	<p>Allows each bit of Port A to be configured for interrupts. By default, the generation of interrupts is disabled. Whenever a 1 is written to a bit of this register, it configures the corresponding bit on Port A to become an interrupt; otherwise, Port A operates as a normal GPIO signal. Interrupts are disabled on the corresponding bits of Port A if the corresponding data direction register is set to Output or if Port A mode is set to Hardware.</p> <p>0 – Configure Port A bit as normal GPIO signal (default) 1 – Configure Port A bit as interrupt</p> <p>Reset Value: 0x0</p>

8.3.3.14 gpio_intmask

- **Name:** Interrupt mask
- **Size:** GPIO_PWIDTH_A
- **Address offset:** 0x34
- **Read/write access:** read/write

This register is available only if Port A is configured to generate interrupts (GPIO_PORTA_INTR = Include (1)).

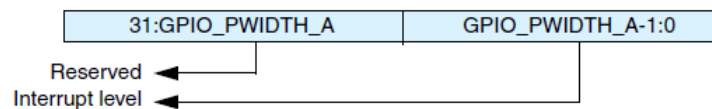


Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	Interrupt mask	R/W	Controls whether an interrupt on Port A can create an interrupt for the interrupt controller by not masking it. By default, all interrupts bits are unmasked. Whenever a 1 is written to a bit in this register, it masks the interrupt generation capability for this signal; otherwise interrupts are allowed through. The unmasked status can be read as well as the resultant status after masking. 0 – Interrupt bits are unmasked (default) 1 – Mask interrupt Reset Value: 0x0

8.3.3.15 gpio_inttype_level

- **Name:** Interrupt level
- **Size:** GPIO_PWIDTH_A
- **Address offset:** 0x38
- **Read/write access:** read/write

This register is available only if Port A is configured to generate interrupts (GPIO_PORTA_INTR = Include (1)).

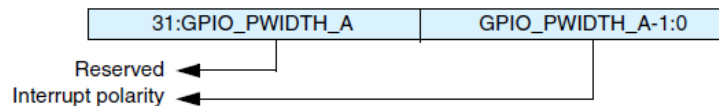


Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	Interrupt level	R/W	Controls the type of interrupt that can occur on Port A. Whenever a 0 is written to a bit of this register, it configures the interrupt type to be level-sensitive; otherwise, it is edge-sensitive. 0 – Level-sensitive (default) 1 – Edge-sensitive Reset Value: 0x0

8.3.3.16 gpio_int_polarity

- **Name:** Interrupt polarity
- **Size:** GPIO_PWIDTH_A
- **Address offset:** 0x3C
- **Read/write access:** read/write

This register is available only if Port A is configured to generate interrupts (GPIO_PORTA_INTR = Include (1)).



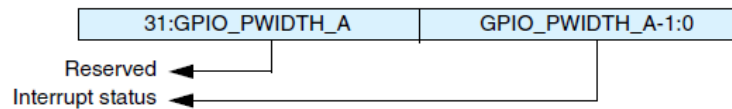
Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	Interrupt polarity	R/W	Controls the polarity of edge or level sensitivity that can occur on input of Port A. Whenever a 0 is written to a bit of this register, it configures the interrupt type to falling-edge or active-low sensitive; otherwise, it is rising-edge or active-high sensitive.

			0 – Active-low (default) 1 – Active-high Reset Value: 0x0
--	--	--	--

8.3.3.17 gpio_intstatus

- **Name:** Interrupt status
- **Size:** GPIO_PWIDTH_A
- **Address offset:** 0x40
- **Read/write access:** read

This register is available only if Port A is configured to generate interrupts (GPIO_PORTA_INTR = Include (1)).

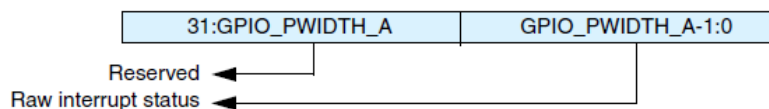


Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	Interrupt status	R	Interrupt status of Port A. Reset Value: 0x0

8.3.3.18 gpio_raw_intstatus

- **Name:** Raw interrupt status
- **Size:** GPIO_PWIDTH_A
- **Address offset:** 0x44
- **Read/write access:** read

This register is available only if Port A is configured to generate interrupts (GPIO_PORTA_INTR = Include (1)).

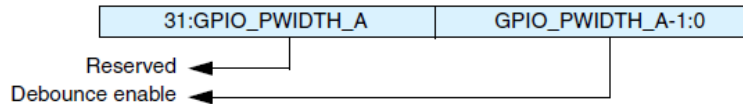


Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	Raw interrupt status	R	Raw interrupt of status of Port A (premasking bits). Reset Value: 0x0

8.3.3.19 gpio_debounce

- **Name:** Debounce enable
- **Size:** GPIO_PWIDTH_A
- **Address offset:** 0x48
- **Read/write access:** read/write

This register is available only if Port A is configured to generate interrupts (GPIO_PORTA_INTR = Include (1)) and when the debounce logic is included (GPIO_DEBOUNCE = Include (1)).

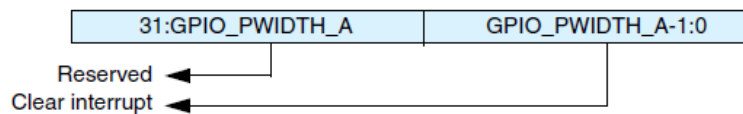


Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	Debounce enable	R/W	Controls whether an external signal that is the source of an interrupt needs to be debounced to remove any spurious glitches. Writing a 1 to a bit in this register enables the debouncing circuitry. A signal must be valid for two periods of an external clock before it is internally processed. 0 – No debounce (default) 1 – Enable debounce Reset Value: 0x0

8.3.3.20 gpio_porta_eoi

- **Name:** Clear interrupt
- **Size:** GPIO_PWIDTH_A
- **Address offset:** 0x4C
- **Read/write access:** write

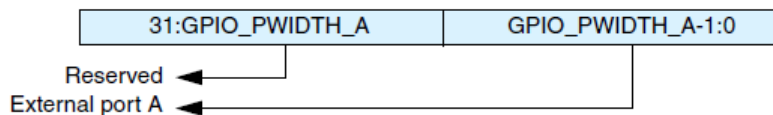
This register is available only if Port A is configured to generate interrupts (GPIO_PORTA_INTR = Include (1)).



Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	Clear interrupt	W	Controls the clearing of edge type interrupts from Port A. When a 1 is written into a corresponding bit of this register, the interrupt is cleared. All interrupts are cleared when Port A is not configured for interrupts. 0 – No interrupt clear (default) 1 – Clear interrupt Reset Value: 0x0

8.3.3.21 gpio_ext_porta

- **Name:** External Port A
- **Size:** GPIO_PWIDTH_A
- **Address offset:** 0x50
- **Read/write access:** read

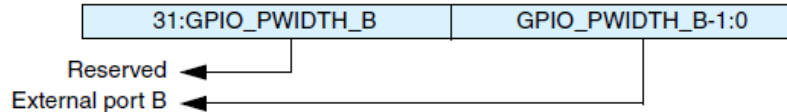


Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved

GPIO_PWIDTH_A-1:0	External Port A	R	When Port A is configured as Input, then reading this location reads the values on the signal. When the data direction of Port A is set as Output, reading this location reads the data register for Port A. Reset Value: 0x0
-------------------	-----------------	---	---

8.3.3.22 gpio_ext_portb

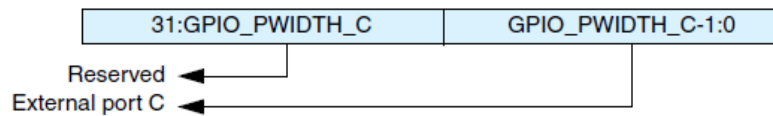
- **Name:** External Port B
- **Size:** GPIO_PWIDTH_B
- **Address offset:** 0x54
- **Read/write access:** read



Bit	Name	Access	Description
31:GPIO_PWIDTH_B	Reserved	Read as zero	Reserved
GPIO_PWIDTH_B-1:0	External Port B	R	When Port B is configured as Input, then reading this location reads the values on the signal. When the data direction of Port B is set as Output, reading this location reads the data register for Port B. Reset Value: 0x0

8.3.3.23 gpio_ext_portc

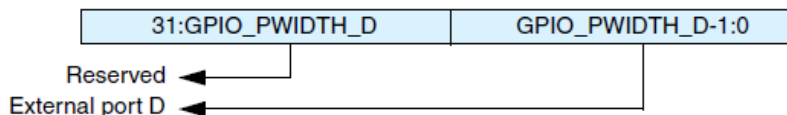
- **Name:** External Port C
- **Size:** GPIO_PWIDTH_C
- **Address offset:** 0x58
- **Read/write access:** read



Bit	Name	Access	Description
31:GPIO_PWIDTH_C	Reserved	Read as zero	Reserved
GPIO_PWIDTH_C-1:0	External Port C	R	When Port C is configured as Input, then reading this location reads the values on the signal. When the data direction of Port C is set as Output, reading this location reads the data register for Port C. Reset Value: 0x0

8.3.3.24 gpio_ext_portd

- **Name:** External Port D
- **Size:** GPIO_PWIDTH_D
- **Address offset:** 0x5C
- **Read/write access:** read

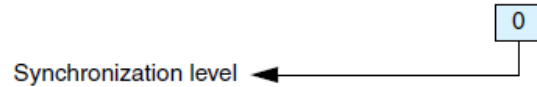


Bit	Name	Access	Description
31:GPIO_PWIDTH_D	Reserved	Read as zero	Reserved

<i>GPIO_PWIDTH_D-1:0</i>	External Port D	R	When Port D is configured as Input, then reading this location reads the values on the signal. When the data direction of Port D is set as Output, reading this location reads the data register for Port D. Reset Value: 0x0
--------------------------	-----------------	---	---

8.3.3.25 gpio_ls_sync

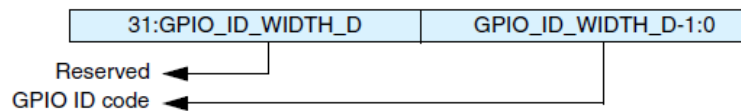
- **Name:** Synchronization level
- **Size:** 1 bit
- **Address offset:** 0x60
- **Read/write access:** read/write



Bit	Name	Access	Description
31:1	Reserved	Read as zero	Reserved
0	Synchronization level	R/W	Writing a 1 to this register results in all level-sensitive interrupts being synchronized to pclk_intr. 0 – No synchronization to pclk_intr (default) 1 – Synchronize to pclk_intr Reset Value: 0x0

8.3.3.26 gpio_id_code

- **Name:** GPIO ID code
- **Size:** GPIO_ID_PWIDTH
- **Address offset:** 0x64
- **Read/write access:** read

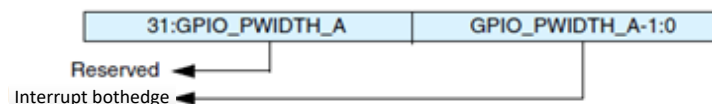


Bit	Name	Access	Description
31: <i>GPIO_ID_PWIDTH</i>	Reserved	Read as zero	Reserved
<i>GPIO_ID_PWIDTH-1:0</i>	GPIO ID code	R	This is a user-specified code that a system can read. It can be used for chip identification, and so on. Reset Value: <i>GPIO_ID_NUM</i>

8.3.3.27 gpio_int_bothedge

- **Name:** GPIO INT Bothedge Type
- **Size:** GPIO_ID_PWIDTH
- **Address offset:** 0x68
- **Read/write access:** read/write

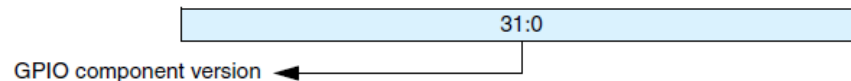
This register is available only if Port A is configured to generate interrupts (GPIO_PORTA_INTR = Include (1)) and interrupt detection is configured to generate on both rising and falling edges of external input signal (GPIO_INT_BOTH_EDGE=Include (1)).



Bit	Name	Access	Description
31:GPIO_PWIDTH_A	Reserved	Read as zero	Reserved
GPIO_PWIDTH_A-1:0	GPIO int bothedge	R/W	This is a user-specified code that a system can read. It can be used for chip identification, and so on. Reset Value: GPIO_ID_NUM

8.3.3.28 gpio_ver_id_code

- **Name:** GPIO Component Version
- **Size:**32 bits
- **Address offset:** 0x6C
- **Read/write access:** read

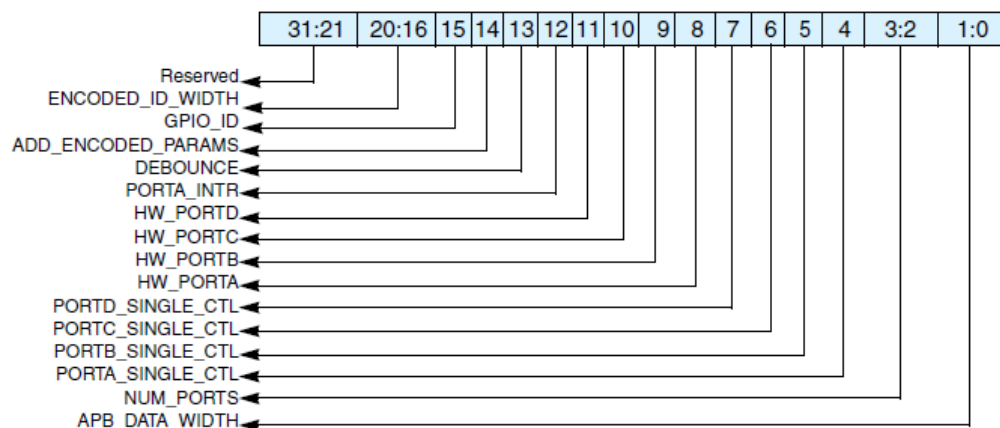


Bit	Name	Access	Description
31:0	GPIO Component Version	R	ASCII value for each number in the version, followed by *. For example 32_30_31_2A represents the version 2.01*. Reset Value: See the releases table in the <i>Release Notes</i>

8.3.3.29 gpio_config_reg1

- **Name:** GPIO Configuration Register 1
- **Size:**32 bits
- **Address offset:** 0x74
- **Read/write access:** read

This register is present when the configuration parameter GPIO_ADD_ENCODED_PARAMS is set to True. If this parameter is set to False, this register reads back zero (0).



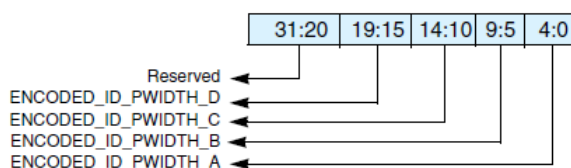
Bit	Name	Access	Description
31:21	Reserved	Read as zero	Reserved
20:16	ENCODED_ID_WIDTH	R	The value of this register is equal to GPIO_ID_WIDTH-1.
15	GPIO_ID	R	The value of this register is derived from the GPIO_ID configuration parameter. 0 = Exclude 1 = Include
14	ADD_ENCODED_PARAMS	R	The value of this register is derived from the GPIO_ADD_ENCODED_PARAMS configuration parameter.

			0 = False 1 = True
13	DEBOUNCE	R	The value of this register is derived from the GPIO_DEBOUNCE configuration parameter. 0 = Exclude 1 = Include
12	PORTA_INTR	R	The value of this register is derived from the GPIO_PORTA_INTR configuration parameter. 0 = Exclude 1 = Include
11	HW_PORTD	R	The value of this register is derived from the GPIO_HW_PORTD configuration parameter. 0 = Exclude 1 = Include
10	HW_PORTC	R	The value of this register is derived from the GPIO_HW_PORTC configuration parameter. 0 = Exclude 1 = Include
9	HW_PORTB	R	The value of this register is derived from the GPIO_HW_PORTB configuration parameter. 0 = Exclude 1 = Include
8	HW_PORTA	R	The value of this register is derived from the GPIO_HW_PORTA configuration parameter. 0 = Exclude 1 = Include
7	PORTD_SINGLE_CTL	R	The value of this register is derived from the GPIO_PORTD_SINGLE_CTL configuration parameter. 0 = False 1 = True
6	PORTC_SINGLE_CTL	R	The value of this register is derived from the GPIO_PORTC_SINGLE_CTL configuration parameter. 0 = False 1 = True
5	PORTB_SINGLE_CTL	R	The value of this register is derived from the GPIO_PORTB_SINGLE_CTL configuration parameter. 0 = False 1 = True
4	PORTA_SINGLE_CTL	R	The value of this register is derived from the GPIO_PORTA_SINGLE_CTL configuration parameter. 0 = False 1 = True
3:2	NUM_PORTS	R	The value of this register is derived from the GPIO_NUM_PORT configuration parameter. 0x0 = 1 0x1 = 2 0x2 = 3 0x3 = 4
1:0	APB_DATA_WIDTH	R	The value of this register is derived from the GPIO_APB_DATA_WIDTH configuration parameter. 0x0 = 8 bits 0x1 = 16 bits 0x2 = 32 bits 0x3 = Reserved

8.3.3.30 gpio_config_reg2

- **Name:** GPIO Configuration Register 1
- **Size:** 32 bits
- **Address offset:** 0x70
- **Read/write access:** read

This register is a read-only register that is present when the configuration parameter GPIO_ADD_ENCODED_PARAMS is set to True. If this configuration is set to False, then this register reads back 0.



Bit	Name	Access	Description
31:20	Reserved	Read as zero	Reserved
19:15	ENCODED_ID_PWIDTH_D	R	The value of this register is equal to GPIO_PWIDTH_D-1.
14:10	ENCODED_ID_PWIDTH_C	R	The value of this register is equal to GPIO_PWIDTH_C-1.
9:5	ENCODED_ID_PWIDTH_B	R	The value of this register is equal to GPIO_PWIDTH_B-1.
4:0	ENCODED_ID_PWIDTH_A	R	The value of this register is equal to GPIO_PWIDTH_A-1.

8.4 Programming the GPIO

GPIO can be programmed via software registers or the GPIO low-level software driver.

8.4.1 Software Registers

The software registers are described in more details in “Registers”.

8.4.2 Programming Considerations

- Reading from an unused location or unused bit sin a particular register always returns zeros. There is no error mechanism in the APB.
- Programming the GPIO registers for interrupt capability, edge-sensitive or level-sensitive interrupts, and interrupt polarity should be completed prior to enabling the interrupts on Port A/B in order to prevent spurious glitches on the interrupt lines to the interrupt controller.
- Writing to the interrupt clear register clears an edge-detected interrupt and has no effect on a level-sensitive interrupt. If, for example, the GPIO_PORTA_INTR configuration parameter is equal to 0, then the following all read back 0:
 - gpio_inten
 - gpio_intmask
 - gpio_int_level
 - gpio_int_polarity
 - gpio_int_status
 - gpio_raw_intstatus
 - gpio_debounce
 - gpio_ls_sync
 - gpio_porta_eoi
- When reading back registers that are no longer present due to configuration parameters settings, then 0 is read back. For example, if APB_DATA_WITDH = 32 bits and GPIO_PWIDTH_A = 8, then the top 24 bits read back 0.

9 Direct Memory Access Controller (DMAC)

9.1 Product Overview

This chapter provides a basic overview of the Ameba-D DMAC, which is an AHB-Central DMA Controller core that transfers data from a source peripheral to a destination peripheral over one or more AHB bus.

9.1.1 General Product Description

Fig 9-1 shows the following functional groupings of the main interfaces to the DMAC block.

- DMA hardware request interface
- Up to eight channels
- FIFO per channel for source and destination
- Arbiter
- AHB master interface
- AHB slave interface

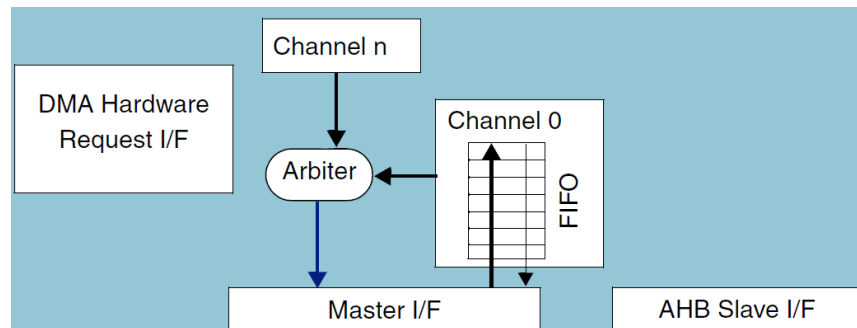


Fig 9-1 Block diagram of DMAC

One channel of the DMAC is required for each source/destination pair. In the most basic configurations, as illustrated in Fig 9-2, the DMAC has one master interface and one channel. The master interface reads the data from a source peripheral (A) and writes it to a destination peripheral (B). Two AHB transfers are required for each DMA data transfer; this is also known as a dual-access transfer.

Fig 9-2 illustrates a peripheral-to-peripheral DMA transfer, where peripheral A (source) uses a hardware handshaking interface, and peripheral B (destination) uses a software handshaking interface. For example, the request to send data to peripheral B is originated by the CPU, while writing to peripheral B is handled by the DMAC. The channel source and destination arbitrate independently for the AHB master interface, along with other channels.

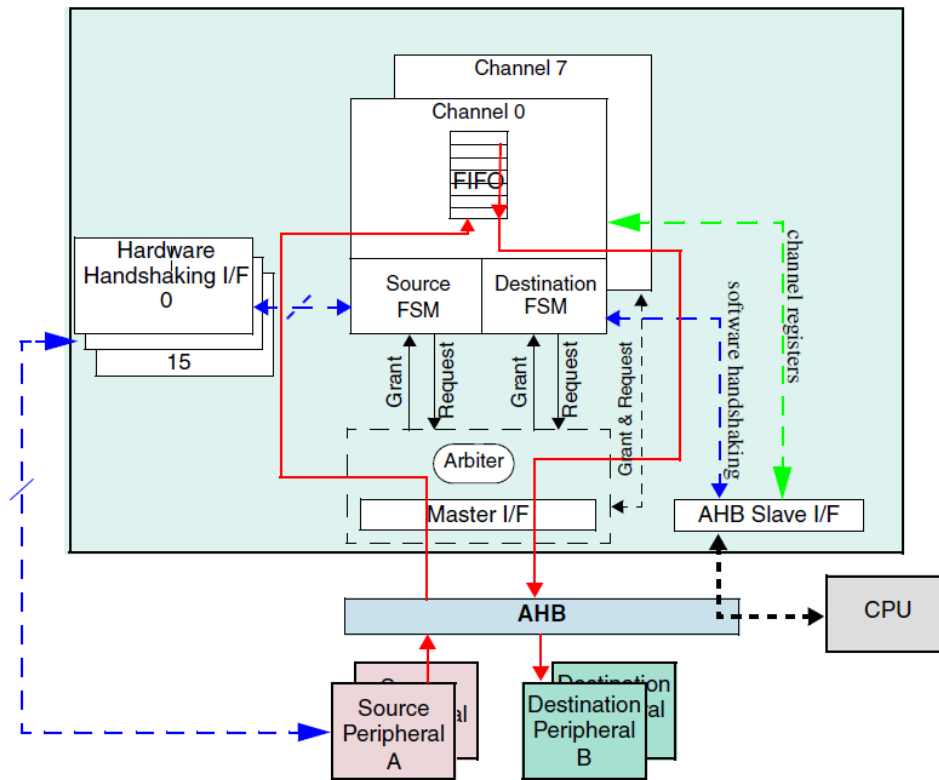


Fig 9-2 Peripheral-to-Peripheral DMA transfer on the same AHB layer

The DMAC also supports multi-layer DMA transfers when the source and destination peripherals are on different AHB layers. In this case, you must configure the DMAC to have more than one master interface—one per layer. Fig 9-3 illustrates a DMAC with two master interfaces and a DMA transfer between a source and destination on different AHB layers. Peripheral B uses a hardware handshaking interface. The memory does not use any handshaking interface to the DMAC in order to initiate DMA transfers.

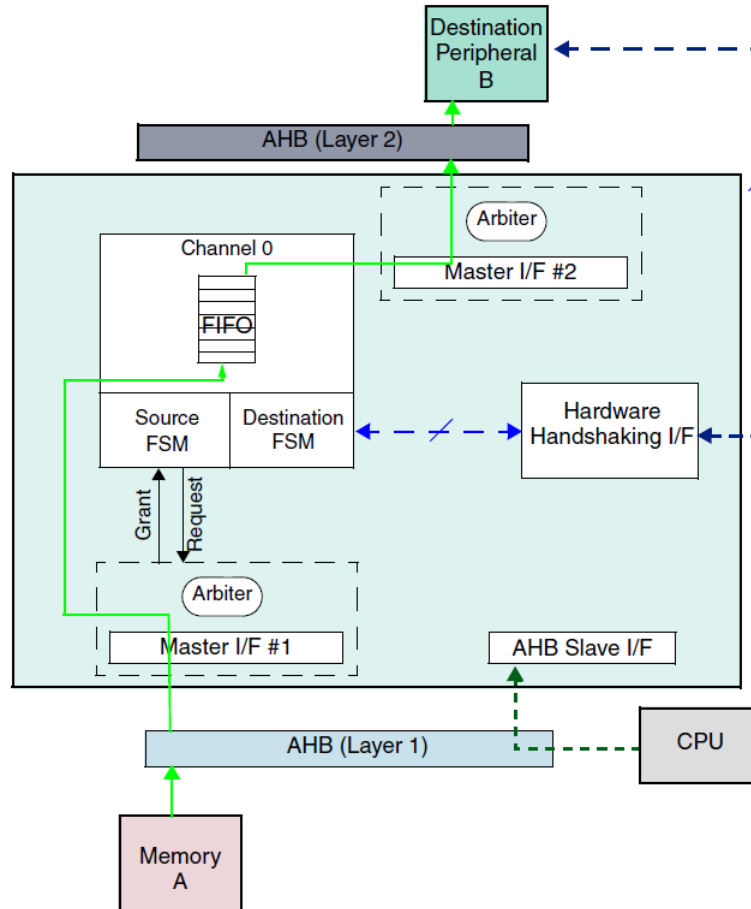


Fig 9-3 Peripheral-to-Memory DMA transfer on separate AHB layers

9.1.2 Basic Definitions

The following terms are concise definitions of the DMA concepts used throughout this databook:

- **Source peripheral** – Device on a AHB layer from which the DMAC reads data; the DMAC then stores the data in the channel FIFO. The source peripheral teams up with a destination peripheral to form a channel. The source peripheral is either an AHB or APB slave. If the source is an APB slave, it is accessed through the AHB-APB bridge.
- **Destination peripheral** – Device to which the DMAC writes the stored data from the FIFO (previously read from the source peripheral). The destination peripheral is either an AHB or APB slave. If the destination is an APB slave, it is accessed through the AHB-APB bridge.
- **Memory** – Source or destination that is always ready for a DMA transfer and does not require a handshaking interface to interact with the DMAC. A peripheral should be assigned as memory only if it does not insert more than 16 wait states. If more than 16 wait states are required, then the peripheral should use a handshaking interface—the default if the peripheral is not programmed to be memory—in order to signal when the peripheral is ready to accept or supply data. A memory peripheral can also generate SPLIT/RETRY responses.
- **Channel** – Read/write data path between a source peripheral on one configured AHB layer and a destination peripheral on the same or different AHB layer that occurs through the channel FIFO. If the source peripheral is not memory, then a source handshaking interface is assigned to the channel. If the destination peripheral is not memory, then a destination handshaking interface is assigned to the channel. Source and destination handshaking interfaces can be assigned dynamically by programming the channel registers.
- **Master interface** – DMAC is a master on the AHB bus, reading data from the source and writing it to the destination over the AHB bus. It is possible to have up to four master interfaces, which means that up to four independent source and destination channels can operate simultaneously. Each channel has to arbitrate for the master interface. You need to have more than one master interface if the source and destination peripherals reside on different AHB layers.
- **Slave interface** – The AHB interface over which the DMAC is programmed. The slave interface in practice can be on the same layer as any of the master interfaces, or it can be on a separate layer.
- **Handshaking interface** – A set of signals or software registers that conform to a protocol and handshake between the DMAC and source or destination peripheral in order to control transferring a single or burst transaction between them. This interface is used to request,

acknowledge, and control a DMAC transaction. A channel can receive a request through one of three types of handshaking interface: hardware, software, or peripheral interrupt.

- **Hardware handshaking interface** – Uses hardware signals to control transferring a single or burst transaction between the DMAC and the source or destination peripheral.
- **Peripheral interrupt handshaking interface** – Simple use of the hardware handshaking interface. In this mode, the interrupt line from the peripheral is tied to the dma_req input of the hardware handshaking interface; other interface signals are ignored.
- **Flow controller** – Device (either the DMAC, or source/destination peripheral) that determines the length of a DMA block transfer and terminates it.
 - If you know the length of a block before enabling the channel, then you should program the DMAC as the flow controller.
 - If the length of a block is not known prior to enabling the channel, the source or destination peripheral needs to terminate a block transfer. In this mode, the peripheral is the flow controller.
- **Flow control mode (CFGx.FCMODE)** – Special mode that only applies when the destination peripheral is the flow controller. It controls the data pre-fetching from the source peripheral.
- **Transfer hierarchy** – Fig 9-4 illustrates the hierarchy between DMA transfers, block transfers, transactions (single or burst), and AHB transfers (single or burst) for non-memory peripherals. Fig 9-5 shows the transfer hierarchy for memory.

Note: For memory peripherals, there is no DMA Transaction Level.

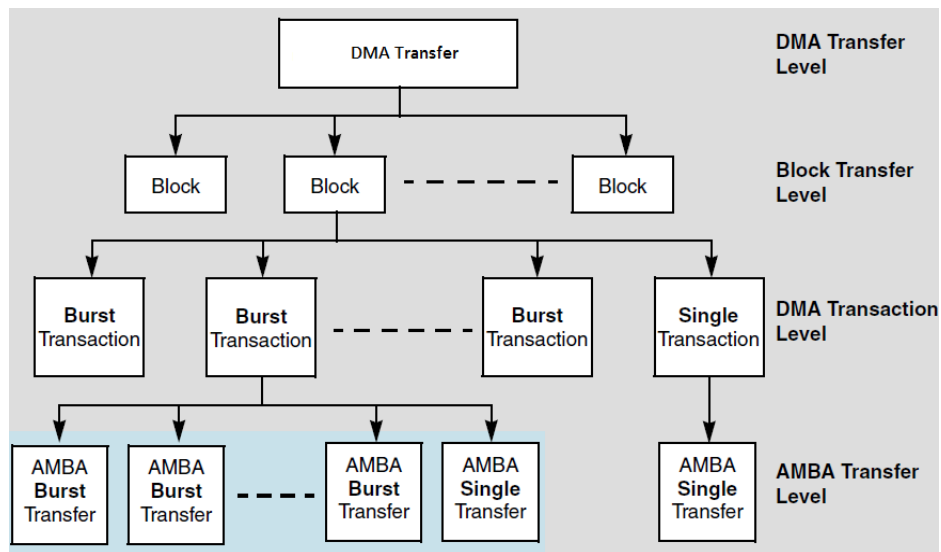


Fig 9-4 DMA Transfer Hierarchy for Non-Memory Peripherals

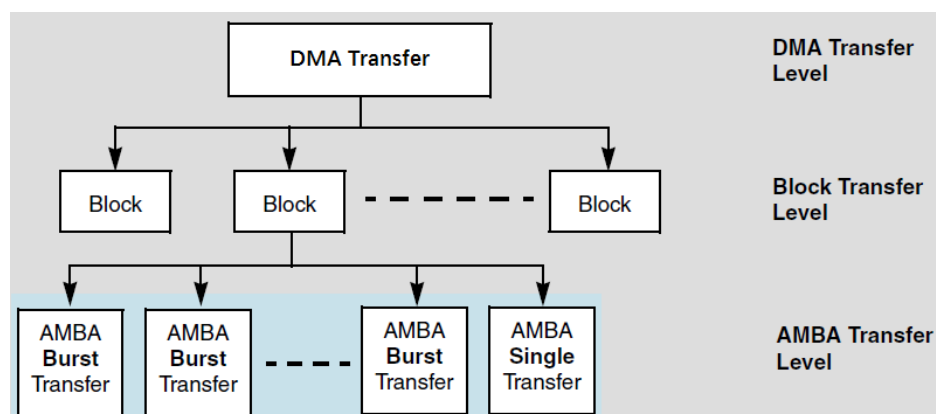


Fig 9-5 DMA transfer hierarchy for memory

- **Block**—Block of DMAC data, the amount of which is the block length and is determined by the flow controller. For transfers between the DMAC and memory, a block is broken directly into a sequence of bursts and single transfers. For transfers between the DMAC and a non-memory peripheral, a block is broken into a sequence of DMAC transactions (single and bursts). These are in turn broken into a sequence of AHB transfers.
- **Transaction** — Basic unit of a DMA transfer. A transaction is relevant only for transfers between the DMAC and a source or destination peripheral if the peripheral is a non-memory device. There are two types of transactions:
 - **Single transaction** — Length of a single transaction is always 1 and is converted to a single AHB transfer.
 - **Burst transaction** — Length of a burst transaction is programmed into the DMAC. The burst transaction is converted into a sequence of bursts and AHB single transfers. DMAC executes each burst transfer by performing incremental bursts that are no longer than the maximum burst size set; the only type of hburst in this kind of transaction is incremental. The burst transaction length is under program control and normally bears some relationship to the FIFO sizes in the DMAC and in the source and destination peripherals.
- **DMA transfer** — Software controls the number of blocks in a DMA transfer. Once the DMA transfer has completed, the hardware within the DMAC disables the channel and can generate an interrupt to signal the DMA transfer completion. You can then reprogram the channel for a new DMA transfer.
 - **Single-block DMA transfer** — Consists of a single block.
 - **Multi-block DMA transfer** — DMA transfer may consist of multiple DMAC blocks. Multi-block DMA transfers are supported through block chaining (linked list pointers), auto-reloading channel registers, and contiguous blocks. The source and destination can independently select which method to use.
 - ◆ **Linked lists (block chaining)** — Linked list pointer (LLP) points to the location in system memory where the next linked list item (LLI) exists. The LLI is a set of registers that describes the next block (block descriptor) and an LLP register. The DMAC fetches the LLI at the beginning of every block when block chaining is enabled.
LLI accesses are always 32-bit accesses (Hsize=2) aligned to 32-bit boundaries and cannot be changed or programmed to anything other than 32-bit, even if the AHB master interface of the LLI supports more than a 32-bit data width.
 - ◆ **Auto-reloading** — DMAC automatically reloads the channel registers at the end of each block to the value when the channel was first enabled.
 - ◆ **Contiguous blocks** — Address between successive blocks is selected to be a continuation from the end of the previous block.
- **Scatter**—Relevant to destination transfers within a block. The destination address is incremented or decremented by a programmed amount when a scatter boundary is reached. The number of AHB transfers between successive scatter boundaries is under software control.
- **Gather** — Relevant to source transfers within a block. The source address is incremented or decremented by a programmed amount when a gather boundary is reached. The number of AHB transfers between successive gather boundaries is under software control.
- **Channel locking** — Software can program a channel to keep the AHB master interface by locking arbitration of the master bus interface for the duration of a DMA transfer, block, or transaction (single or burst).
- **Bus locking** — Software can program a channel to maintain control of the AHB bus by asserting hlock for the duration of a DMA transfer, block, or transaction (single or burst). At minimum, channel locking is asserted during bus locking.
- **FIFO mode** — Special mode to improve bandwidth. When enabled, the channel waits until the FIFO is less than half full to fetch the data from the source peripheral, and waits until the FIFO is greater than or equal to half full in order to send data to the destination peripheral. Because of this, the channel can transfer the data using bursts, which eliminates the need to arbitrate for the AHB master interface in each single AHB transfer. When this mode is not enabled, the channel waits only until the FIFO can transmit or accept a single AHB transfer before it requests the master bus interface.
- **Pseudo fly-by operation** — Typically, it takes two AHB bus cycles to complete a transfer—one for reading the source and one for writing to the destination. However, when the source and destination peripherals of a DMA transfer are on different AHB layers, it is possible for the DMAC to fetch data from the source and store it in the channel FIFO at the same time that the DMAC extracts data from the channel FIFO and writes it to the destination peripheral. This activity is known as *pseudofly-by operation*. In order for this to occur in appropriate sequential order, the source and destination logic in the DMAC should first win their respective master interfaces, and then the master interface for both source and destination layers must win arbitration of their AHB layer.

9.1.3 Features

The DMAC component includes the following features.

9.1.3.1 General

- AMBA2.0-compliant
- AHB slave interface — used to program the DMAC
- Channels
 - Up to eight channels, one per source and destination pair
 - Unidirectional channels — data transfers in one direction only

- Programmable channel priority
- AHB master interface(s)
 - Up to four independent AHB master interfaces that allows:
 - ◆ Up to four simultaneous DMA transfers
 - ◆ Masters that can be on different AHB layers (multi-layer support)
 - ◆ Source and destination that can be on different AHB layers (pseudo fly-by performance)
 - Configurable data bus width (up to 256 bits) for each AHB master interface
 - Configurable endianness for master interfaces
- Transfers
 - Support for memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral DMA transfers
 - DMAC to or from APB peripherals through the APB bridge
- Configurable identification register
- Component ID parameters for configurable software driver support
- Configuration of AHB Lite system

9.1.3.2 Address Generation

- Programmable source and destination addresses (on AHB bus)
- Address increment, decrement, or no change
- Multi-block transfers achieved through:
 - Linked Lists (block chaining)
 - Auto-reloading of channel registers
 - Contiguous address between blocks
- Independent source and destination selection of multi-block transfer type
- Scatter/Gather

9.1.3.3 Channel Buffering

- Single FIFO per channel for source and destination
- Configurable FIFO depth
- D flip-flop-based FIFO
- Automatic data packing or unpacking to fit FIFO width

9.1.3.4 Channel Control

- Programmable source and destination for each channel
- Programmable transfer type for each channel (memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral)
- Programmable burst transaction size for each channel
- Programmable enable and disable of DMA channel
- Support for disabling channel without data loss
- Support for suspension of DMA operation
- Support for RETRY, SPLIT, and ERROR responses
- Programmable maximum burst transfer size per channel
- Configurable maximum transaction size to allow gate optimization
- Configurable maximum block size to allow gate optimization
- Bus locking – can be programmed to be over the transaction, block, or DMA transfer level
- Channel locking – can be programmed to be over the transaction, block, or DMA transfer level
- Option to hardcode type of multi-block transfer
- Option to disable the write back of the Channel Control register at the end of every block transfer

9.1.3.5 Transfer Initiation

- Handshaking interfaces for source and destination peripherals (up to 16)
 - Hardware handshaking interface (software handshaking interface not supported)
 - Peripheral interrupt handshaking interface

- Handshaking interface supports single or burst DMA transactions
- Polarity control for hardware handshaking interface
- Enabling and disabling of individual DMA handshaking interfaces

9.1.3.6 Flow Control

- Programmable flow control at block transfer level (source, destination, or DMAC)
- Software control of source data pre-fetch when destination is flow controller

9.1.3.7 Interrupts

- Combined and separate interrupt requests
- Interrupt generation on:
 - DMA transfer (multi-block) completion
 - Block transfer completion
 - Single and burst transaction completion
 - Error condition
- Support of interrupt enabling and masking

9.2 Functional Description

This chapter describes the functional details of the DMAC component. There is an option to configure AHB Lite, which is the implementation of AMBA 2.0 AHB-Lite. The AHB Lite configuration does not include the following:

- Requesting/granting protocols to the arbiter and split/retry responses from the slaves; all slaves are made non-split capable
- No arbiter as the signals associated with the component are not used: hbusreq and hgrant
- No write data, address, or control multiplexers
- Pause mode not enabled
- Default master number changed to 1
- Number of masters is changed to 1

9.2.1 Setup/Operation of DMA Transfers

“Programming a Channel” describes how to program the DMAC in order to perform DMA transfers. This section discusses how a single block transfer, made up of transactions, is actually performed. The relevant settings of the DMAC are also discussed here.

9.2.2 Block Flow Controller and Transfer Type

The device that controls the length of a block is known as the flow controller. Either the DMAC, the source peripheral, or the destination peripheral must be assigned as the flow controller.

- If the block size is known prior to when the channel is enabled, then the DMAC should be programmed as the flow controller. The block size should be programmed into the CTLx.BLOCK_TS field.
- If the block size is unknown when the DMAC channel is enabled, either the source or destination peripheral must be the flow controller.

The CTLx.TT_FC field indicates the transfer type and flow controller for that channel. Table 9-1 lists valid transfer types and flow controller combinations.

Table 9-1 Transfer types and flow control combinations

Transfer Type	Flow Controller
Memory to Memory	DMAC
Memory to Peripheral	DMAC
Memory to Peripheral	Peripheral
Peripheral to Memory	DMAC
Peripheral to Memory	Peripheral
Peripheral to Peripheral	DMAC

Peripheral to Peripheral	Source Peripheral
Peripheral to Peripheral	Destination Peripheral

As an example, the DMAC can be programmed as the flow controller when a DMA block must be transferred from a receive SSI peripheral to memory. In a block transfer, software programs the SSI register – CTRLR1.NDF – with the number of source data items minus 1. Software then programs the CTLx.BLOCK_TS register with the same value and programs the DMAC as the flow controller.

The SSI has no built-in intelligence to signal block completion to the DMAC; this is not required in this case because software knows the block size prior to enabling the channel.

As another example, a peripheral can be a block flow controller when a DMA block must be transferred from an Ethernet controller to memory. In this case, the size of an ethernet packet may not be known prior to enabling the DMAC channel. Therefore, the ethernet controller needs built-in intelligence to indicate to the DMAC when a block transfer has completed.

9.2.3 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or burst transactions. The operation of the handshaking interface is different and depends on whether the peripheral or the DMAC is the flow controller.

The peripheral uses the handshaking interface to indicate to the DMAC that it is ready to transfer or accept data over the AHB bus.

A non-memory peripheral can request a DMA transfer through the DMAC using one of two types of handshaking interfaces:

- Hardware
- Software

Software selects between the hardware or software handshaking interface on a per-channel basis. Software handshaking is accomplished through memory-mapped registers, while hardware handshaking is accomplished using a dedicated handshaking interface.

Note: Throughout the remainder of this document, references to both source and destination hardware handshaking interfaces assume an active-high interface (refer to CFGx.SRC(DST)_HS_POL bits in the Channel Configuration register). When active-low handshaking interfaces are used, then the active level and edge are reversed from that of an active-high interface.

The type of handshaking interface depends on whether the peripheral is a flow controller or not.

Note: Source and destination peripherals can independently select the handshaking interface type; that is, hardware or software handshaking. For more information, refer to the CFGx.HS_SEL_SRC and CFGx.HS_SEL_DST parameters in the CFGx register.

9.2.4 Basic Interface Definitions

Note: In this chapter and the following equations, references to CTLx.SRC_MSIZ, CTLx.DEST_MSIZ, CTLx.SRC_TR_WIDTH, and CTLx.DST_TR_WIDTH refer to the decoded values of the parameters; for example, CTLx.SRC_MSIZ = 3'b001 decodes to 4, and CTLx.SRC_TR_WIDTH = 3'b010 decodes to 32 bits.

The following definitions are used in this chapter:

- Source single transaction size in bytes

$$src_single_size_bytes = CTLx.SRC_TR_WIDTH/8 \quad (1)$$
- Source burst transaction size in bytes

$$src_burst_size_bytes = CTLx.SRC_MSIZ * src_single_size_bytes \quad (2)$$
- Destination single transaction size in bytes

$$dst_single_size_bytes = CTLx.DST_TR_WIDTH/8 \quad (3)$$
- Destination burst transaction size in bytes

$$dst_burst_size_bytes = CTLx.DEST_MSIZ * dst_single_size_bytes \quad (4)$$
- Block size in bytes:
 - DMAC is flow controller—With the DMAC as the flow controller, the processor programs the DMAC with the number of data items (block size) of source transfer width (CTLx.SRC_TR_WIDTH) to be transferred by the DMAC in a block transfer; this is programmed into the CTLx.BLOCK_TS field. Therefore, the total number of bytes to be transferred in a block is:

$$blk_size_bytes_dma = CTLx.BLOCK_TS * src_single_size_bytes \quad (5)$$

- Source peripheral is block flow controller

$$blk_size_bytes_src = (\text{Number of source burst transactions in block} * src_burst_size_bytes) + (\text{Number of source single transactions in block} * src_single_size_bytes) \quad (6)$$

- Destination peripheral is block flow controller

$$blk_size_bytes_dst = (\text{Number of destination burst transactions in block} * dst_burst_size_bytes) + (\text{Number of destination single transactions in block} * dst_single_size_bytes) \quad (7)$$

9.2.5 Memory Peripherals

Fig 9-5 shows the DMA transfer hierarchy of the DMAC for a memory peripheral. There is no handshaking interface with the DMAC, and therefore the memory peripheral can never be a flow controller. Once the channel is enabled, the transfer proceeds immediately without waiting for a transaction request.

The alternative to not having a transaction-level handshaking interface is to allow the DMAC to attempt AHB transfers to the peripheral once the channel is enabled. If the peripheral slave cannot accept these AHB transfers, it inserts wait states onto the bus (by de-asserting hready) until it is ready; it is not recommended that more than 16 wait states be inserted onto the bus. By using the handshaking interface, the peripheral can signal to the DMAC that it is ready to transmit or receive data, and then the DMAC can access the peripheral without the peripheral inserting wait states onto the bus.

Note: If a channel is used exclusively for memory-to-memory DMA transfers – that is, no transaction-level handshaking on the source or destination side – then set DMAH_MAX_MULT_SIZE to 4 in order to achieve logic optimization.

9.2.6 Handshaking Interface – Peripheral is Not Flow Controller

When the peripheral is not the flow controller, the DMAC tries to efficiently transfer the data using as little of the bus bandwidth as possible. Generally, the DMAC tries to transfer the data using burst transactions and, where possible, fill or empty the channel FIFO in single bursts – provided that the software has not limited the burst length. The DMAC can also lock the arbitration for the master bus interface so that a channel is permanently granted the master bus interface. Additionally, the DMAC can assert the AMBA hlock signal to lock the AHB system arbiter.

Before describing the handshaking interface operation when the peripheral is not the flow controller, the following sections define the terms “Single Transaction Region” and “Early-Terminated Burst Transaction.”

9.2.6.1 Single Transaction Region

There are cases where a DMA block transfer cannot complete using only burst transactions. Typically, this occurs when the block size is not a multiple of the burst transaction length. In these cases, the block transfer uses burst transactions up to the point where the amount of data left to complete the block is less than the amount of data in a burst transaction. At this point, the DMAC samples the “single” status flag and completes the block transfer using single transactions.

The peripheral asserts a single status flag to indicate to the DMAC that there is enough data or space to complete a single transaction from or to the source/destination peripheral.

Note: For hardware handshaking, the single status flag is a signal on the hardware handshaking interface.

The Single Transaction Region is the time interval where the DMAC uses single transactions to complete the block transfer; burst transactions are exclusively used outside this region.

Note: Burst transactions can also be used in this region.

The Single Transaction Region applies to only a peripheral that is *not* the flow controller. The precise definition of when this region is entered is dependent on what acts as the flow controller:

- The DMAC is the flow controller – The source peripheral enters the Single Transaction Region when the number of bytes left to complete in the source block transfer is less than *src_burst_size_bytes*. If:

$$blk_size_bytes/src_burst_size_bytes = integer \quad (8)$$

then the source never enters this region, and the source block uses only burst transactions.

The destination peripheral enters the Single Transaction Region when the number of bytes left to complete in the destination block transfer is less than *dst_burst_size_bytes*. If:

$$\text{blk_size_bytes}/\text{dst_burst_size_bytes} = \text{integer} \quad (9)$$

then the destination never enters this region, and the destination block uses only burst transactions.

Note: The above conditions cause a peripheral to enter the Single Transaction Region. When the peripheral is outside the Single Transaction Region, then the DMAC responds to only burst transaction requests. Whether the peripheral knows that it is in the Single Transaction Region or not, it must always generate burst requests outside the Single Transaction Region, or the DMA block transfer stalls. Once in the Single Transaction Region, the DMAC can complete the block transfer using single transactions.

- Either the source or destination peripheral is the flow controller – The destination or source peripheral enters the Single Transaction Region when the flow control peripheral—that is, the source or destination – signals the last transaction in the block *and* when the amount of data left to be transferred in the destination/source block is less than that which is specified by *dst_burst_size_bytes/src_burst_size_bytes*.

9.2.6.2 Early-Terminated Burst Transaction

When a source or destination peripheral is in the Single Transaction Region, a burst transaction can still be requested. However, *src_burst_size_bytes/dst_burst_size_bytes* is greater than the number of bytes left to complete in the source/destination block transfer at the time that the burst transaction is triggered. In this case, the burst transaction is started and “early-terminated” at block completion without transferring the programmed amount of data – that is, *src_burst_size_bytes* or *dst_burst_size_bytes* – but only the amount required to complete the block transfer. An Early-Terminated Burst Transaction occurs between the DMAC and the peripheral only when the peripheral is not the flow controller.

9.2.6.3 Hardware Handshaking – Peripheral Is Not Flow Controller

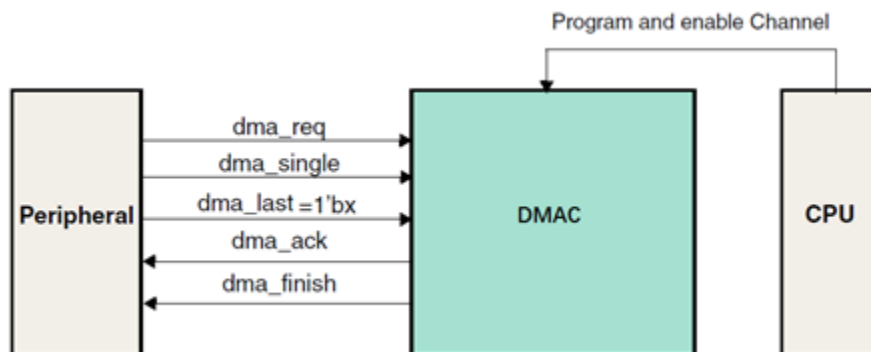


Fig 9-6 illustrates the hardware handshaking interface between a peripheral – whether a destination or source – and the DMAC when the peripheral is not the flow controller.

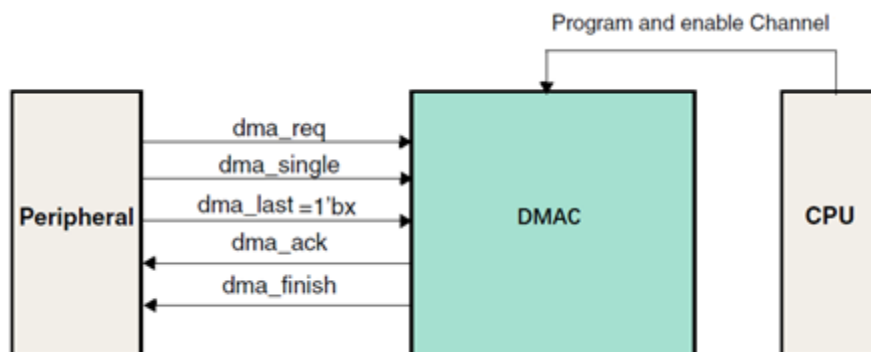


Fig 9-6 Hardware handshaking interface

Table 9-2 describes the hardware handshaking signals in the case where the peripheral is not the flow controller; that is, where either the DMAC or the other peripheral is the flow controller. Signal polarity can be programmed using the CFGx.SRC_HS_POL and CFGx.DST_HS_POL fields.

Table 9-2 Hardware handshaking interface

Signal	Direction	Description
dma_ack	Output	DMAC acknowledge signal to peripheral. The dma_ack signal is asserted after the data phase of the last AHB transfer in the current transaction – single or burst – to the peripheral that has completed. For a single transaction, dma_ack remains asserted until the peripheral de-asserts dma_single; dma_ack is de-asserted one hclk cycle later. For a burst transaction, dma_ack remains asserted until the peripheral de-asserts dma_req; dma_ack is de-asserted one hclk cycle later.
dma_finish	Output	DMAC asserts dma_finish to signal block completion. This has the same timing as dma_ack and forms a handshaking loop with dma_req if the last transaction in the block was a burst transaction, or with dma_single if the last transaction in the block was a single transaction. There is an exception to the above timing definition when dma_finish interfaces with a source peripheral when the destination peripheral is the flow controller.
dma_last	Input	Since the peripheral is not the flow controller, dma_last is not sampled by the DMAC and this signal is ignored.
dma_req	Input	Burst transaction request from peripheral. The DMAC always interprets the dma_req signal as a burst transaction request, regardless of the level of dma_single. This is a level-sensitive signal; once asserted by the peripheral, dma_req must remain asserted until the DMAC asserts dma_ack. Upon receiving the dma_ack signal from the DMAC to indicate the burst transaction is complete, the peripheral should de-assert the burst request signal, dma_req. Once dma_req is de-asserted by the peripheral, the DMAC de-asserts dma_ack. If an active level on dma_req is detected in the Single Transaction Region, then the block is completed using an Early-Terminated Burst Transaction.
dma_single	Input	Single transfer status. The dma_single signal is a status signal that is asserted by a destination peripheral when it can accept at least one destination data item; otherwise it is cleared. For a source peripheral, the dma_single signal is again a status signal and is asserted by a source peripheral when it can transmit at least one source data item; otherwise it is cleared. Once asserted, dma_single must remain asserted until dma_ack is asserted, at which time the peripheral should de-assert dma_single. This signal is sampled by the DMAC only in the Single Transaction Region of the block transfer. Outside of this region, dma_single is ignored and all transactions are burst transactions.

Fig 9-7 shows the timing diagram of a burst transaction where the peripheral clock, per_clk, equals hclk. In this example, the peripheral is outside the Single Transaction Region, and therefore the DMAC does not sample dma_single[0].

The handshaking loop is as follows:

- dma_req asserted by peripheral
- > dma_ack asserted by DW_ahb_dmac
- > dma_req de-asserted by peripheral
- > dma_ack de-asserted by DW_ahb_dmac

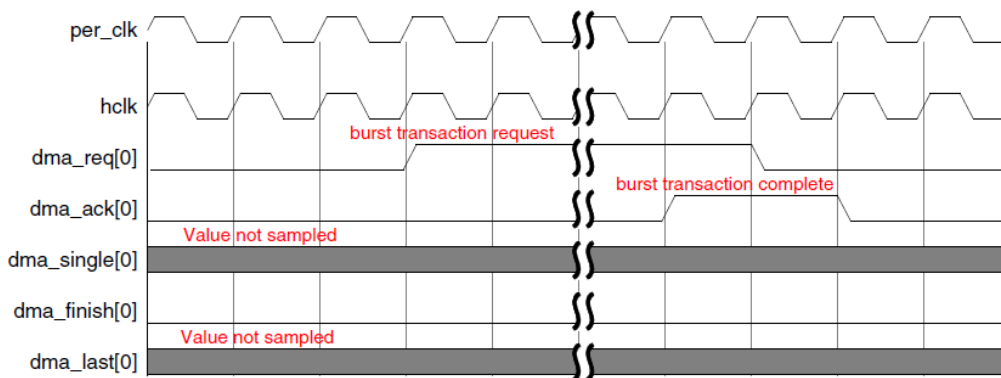


Fig 9-7 Burst transaction – pclk = hclk

The `per_clk` signal is equal to `hclk` if the peripheral is an AHB peripheral; it is equal to `pclk` if the peripheral is an APB peripheral. The burst transaction request signal, `dma_req`, and the single status signal, `dma_single`, are generated in the peripheral of `per_clk` and sampled by `hclk` in the DMAC. The acknowledge signal, `dma_ack`, is generated in the DMAC of `hclk` and sampled in the peripheral by `per_clk`. The handshaking mechanism between the DMAC and the peripheral supports quasi-synchronous clocks; that is, `hclk` and `per_clk` must be phase-aligned, and the `hclk` frequency must be a multiple of the `per_clk` frequency.

In the case where the destination peripheral is an APB peripheral and for the case of buffered writes through an APB bridge, then caution must be exercised so as not to overflow the destination peripheral FIFO. This can happen when the write is buffered in the APB bridge; that is, the write completes on the AHB before completing on the APB bus. The following scenario can cause an overflow: the DMAC asserts `dma_ack` as soon as the write transaction completes on the AHB. The APB peripheral, on sampling that the acknowledge signal is asserted, de-asserts its request signal and asserts the request signal one APB clock cycle later, as it senses that there is space in its FIFO. The issue here is that there could be space for only a single entry that the first buffered write will consume. The initiation of the second transaction may overflow the FIFO. If the write were not buffered, then the initiation of the second transaction would not occur, as the destination peripheral would sense that its FIFO was full.

To avoid this, do one of the following:

- Do not use buffered writes.
- If using buffered writes, then ensure that `dma_ack` signal from the DMAC is delayed until the write completes to the APB peripheral. One way may be to route the `dma_ack` signal through the APB bridge.

Fig 9-8 shows two back-to-back burst transactions at the end of a block transfer where the `hclk` frequency is twice the `pclk` frequency; the peripheral is an APB peripheral. The second burst transaction terminates the block, and `dma_finish[0]` is asserted to indicate block completion.

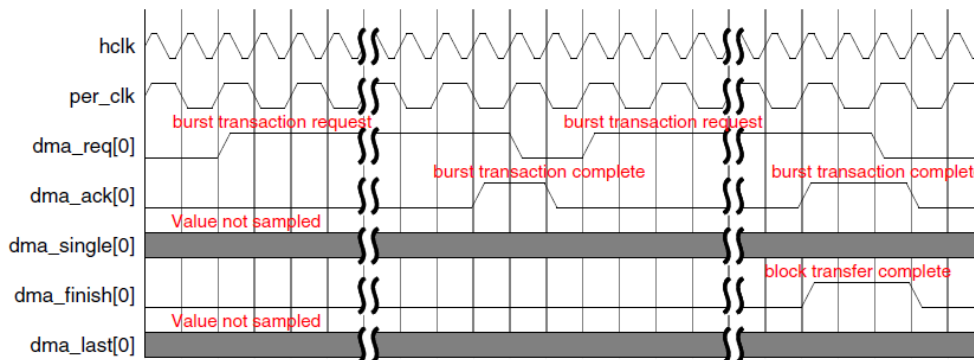


Fig 9-8 Back-to-Back burst transactions – $hclk = 2 * per_clk$

There are two things to note when designing the hardware handshaking interface:

- Once asserted, the `dma_req` burst request signal must remain asserted until the corresponding `dma_ack` signal is received, even if the condition that generates `dma_req` in the peripheral is False.
- The `dma_req` signal should be de-asserted when `dma_ack` is asserted, even if the condition that generates `dma_req` in the peripheral is True.

Fig 9-9 shows a single transaction that occurs in the Single Transaction Region. The handshaking loop is as follows:

```

dma_single asserted by peripheral
-> dma_ack asserted by DW_ahb_dmac
-> dma_single de-asserted by peripheral
-> dma_ack de-asserted by DW_ahb_dmac

```

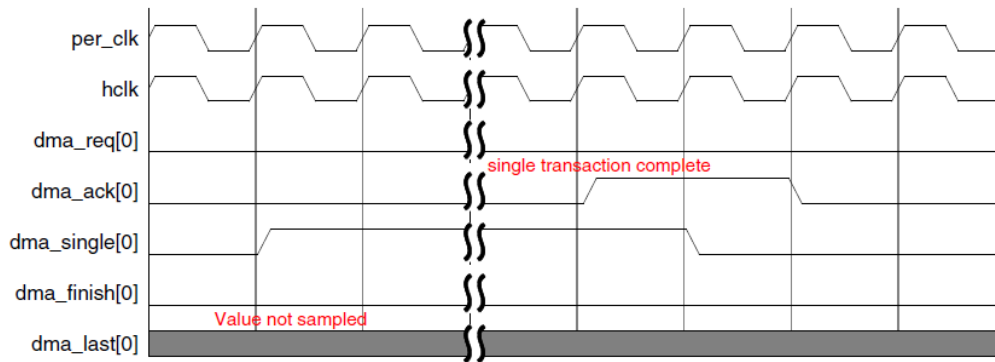


Fig 9-9 Single transaction

Fig 9-10 shows a burst transaction, followed by two back-to-back single transactions, where the hclk frequency is twice the per_clk frequency; handshaking interface 0 is used. After the first burst transaction, the peripheral enters the Single Transaction Region and the DMAC starts sampling dma_single[0]. On hclk edges T2 and T4, the DMAC samples that dma_single[0] is asserted and performs single transactions. The second single transaction terminates the block transfer; dma_finish[0] is asserted to indicate block completion.

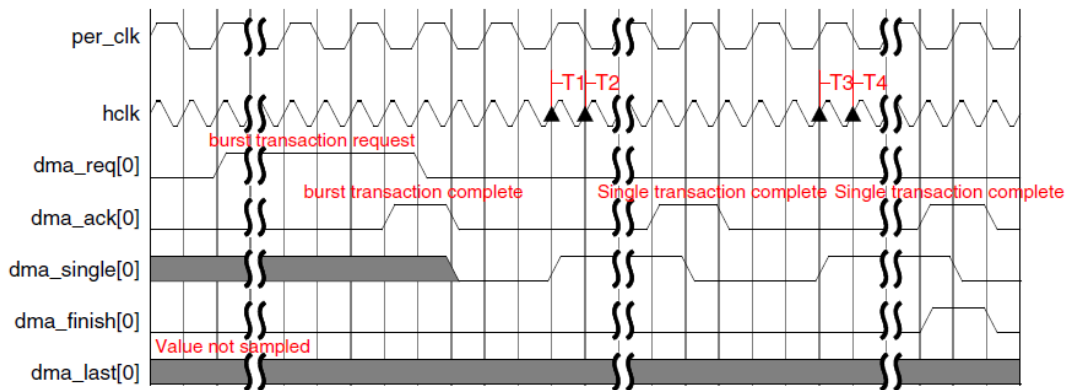


Fig 9-10 Burst followed by Back-to-Back single transactions

In the Single Transaction Region, if an active level on dma_req and dma_single occur on the same cycle – or if the active level on dma_single occurs only one cycle before an active level on dma_req – then the burst transaction takes precedence over the single transaction, and the block would be completed using an Early-Terminated Burst Transaction. If the DMAC samples that dma_req[0] was asserted on hclk cycles T1-T2 or T3-T4 in Fig 9-10, then the burst request takes precedence.

In Fig 9-11, an active level on dma_req[0] after time T4 takes precedence over the active level on dma_single[0] after time T3.

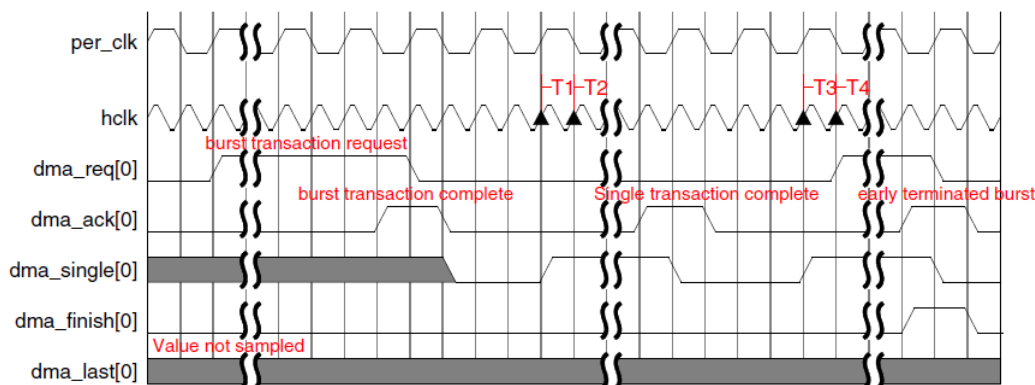


Fig 9-11 Early-Terminated burst transaction

Fig 9-12 shows a burst transaction followed by a single transaction, followed by a burst transaction at the end of a block. After the first burst transaction completes, the peripheral is in the Single Transaction Region and DMAC samples that `dma_single[0]` is asserted at T1. The `dma_req[0]` signal is triggered in the middle of this single transaction at time T2. This burst transaction request is ignored and is not serviced. An active edge on `dma_req[0]` is re-generated and sampled by DMAC at time T3. This burst transaction completes the block transfer using an Early-Terminated Burst Transaction.

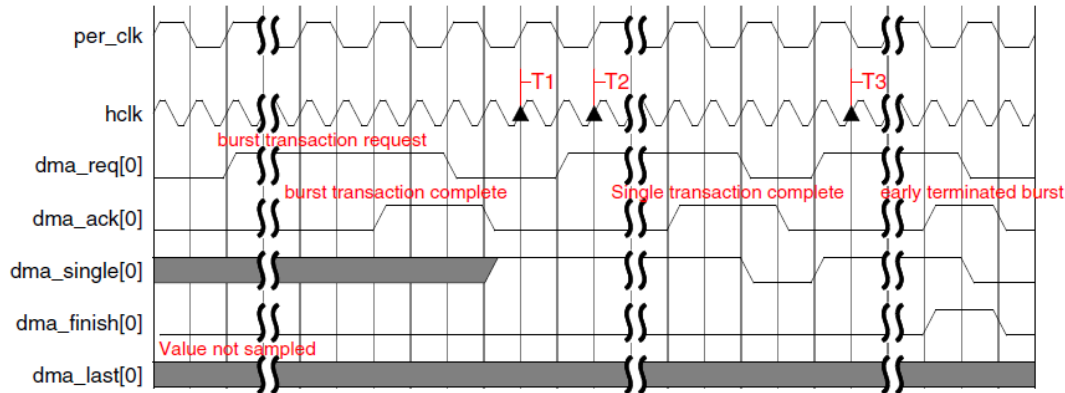


Fig 9-12 Burst transaction ignored during active single transaction

9.2.6.3.1 Generating `dma_req` and `dma_single` Hardware Handshaking Signals

Fig 9-13 illustrates a suggested method of generating `dma_req` and `dma_single` for a source peripheral when the peripheral is not the flow controller. The `single_flag` signal in Fig 9-13 is asserted when the source FIFO has at least one source data item in the FIFO. The `burst_flag` signal in Fig 9-13 is asserted when the source FIFO contains data items greater than or equal to some watermark-level number of data items in it.

Note: Fig 9-13 shows `dma_req` and `dma_single` being de-asserted when `dma_ack` is asserted. It also shows how, once asserted, `dma_req` and `dma_single` remain asserted until `dma_ack` is asserted. The example assumes active-high handshaking.

The destination peripheral `dma_req` and `dma_single` signals can be generated in a similar fashion, but in this case the `single_flag` signal in Fig 9-13 is asserted when the destination FIFO has at least one free location. The `burst_flag` signal in Fig 9-13 is asserted when the destination FIFO contains free locations greater than or equal to some watermark-level number.

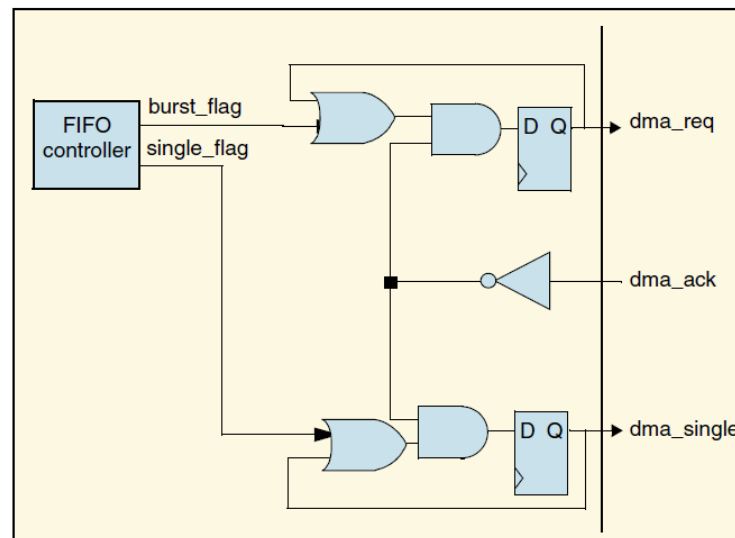


Fig 9-13 Generation of `dma_req` and `dma_single` by source

9.2.6.4 Single Transactions – Peripheral Is Not Flow Controller

When the source peripheral is not the flow controller, the source peripheral can hardcode `dma_single` to an inactive level (hardware handshaking). This can happen if either of the following is true:

- Block size is a multiple of the burst transaction length.
 - If DMAC is the flow controller

$$\text{blk_size_bytes_dma/src_burst_size_bytes} = \text{integer}$$
 - If the destination peripheral is the flow controller

$$\text{blk_size_bytes_dst/src_burst_size_bytes} = \text{integer}$$
- Block size is not a multiple of the burst transaction length, but the peripheral can dynamically adjust the watermark level that triggers a burst request in order to enable block completion.

When the destination peripheral is not a flow controller, then the destination peripheral may hardcode `dma_single` to an inactive level (hardware handshaking). This can happen when any of the following are true:

- Block size is a multiple of the burst transaction length.
 - If DMAC is the flow controller

$$\text{block_size_bytes_dma/dst_burst_size_bytes} = \text{integer}$$
 - If the source peripheral is flow controller

$$\text{block_size_bytes_src/dst_burst_size_bytes} = \text{integer}$$
- The destination peripheral can dynamically adjust the watermark level upwards so that a burst request is triggered in order to enable a destination block completion.
- It is guaranteed that data at some point will be extracted from the destination FIFO in the “Single transaction region” in order to trigger a burst transaction.

Note: The destination peripheral requires data to be extracted in order to empty the destination FIFO below a watermark level for triggering a destination burst request. If it is guaranteed that data at some point will be extracted from the destination FIFO in the Single Transaction Region in order to trigger a `dma_req`, then in this case, the `dma_single` signal from the destination can be tied to an inactive level, and the destination block completes with an Early-Terminated Burst Transaction.

If none of the above are true, then a series of burst transactions followed by single transactions is needed to complete the source/destination block transfer.

9.2.7 Handshaking Interface – Peripheral Is Flow Controller

When the peripheral is the flow controller, it controls the length of the block and must communicate to the DMAC when the block transfer is complete. The peripheral does this by telling the DMAC that the current transaction – burst or single – is the last transaction in the block. When the peripheral is the flow controller and the block size is not a multiple of the `CTLx.SRC_MSIZ/CTLx.DEST_MSIZ`, then the peripheral must use single transactions to complete a block transfer.

Note: Since the peripheral can terminate the block on a single transaction, there is no notion of a Single Transaction Region such as there is when the peripheral is not the flow controller.

When the peripheral is the flow controller, it indicates directly to DMAC which type of transaction – single or burst – to perform. Where possible, the DMAC uses the maximum possible burst length. It can also lock the arbitration for the master bus so that a channel is permanently granted the master bus interface. The DMAC can also assert the `hlock` signal to lock the AHB system arbiter.

9.2.7.1 Hardware Handshaking – Peripheral Is Flow Controller

Fig 9-14 shows the hardware handshaking interface between a destination or source peripheral and the DMAC when the peripheral is the flow controller.

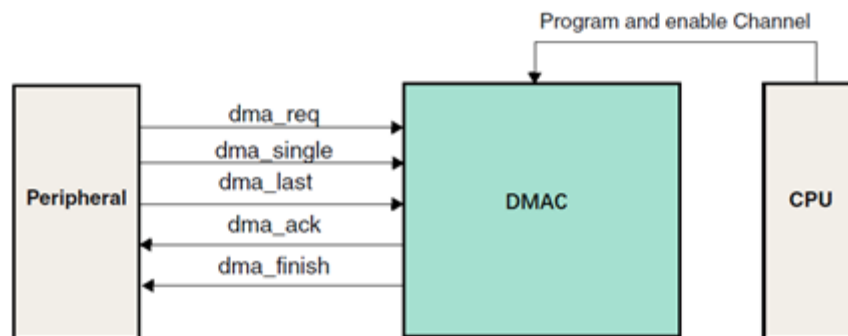


Fig 9-14 Hardware handshaking interface

Table 9-3 describes the operation of the hardware handshaking interface signals when the peripheral is the flow controller; timing diagrams are illustrated in Fig 9-15 and Fig 9-16.

Table 9-3 Hardware handshaking interface

Signal	Direction	Description
dma_ack	Output	DMAC acknowledge signal to the peripheral. This is asserted after the data phase of the last AHB transfer in the current transaction (single or burst) to the peripheral has completed. It forms a handshaking loop with dma_req and remains asserted until the peripheral de-asserts dma_req (de-asserted one hclk cycle later).
dma_finish	Output	DMAC block transfer complete signal. The DMAC asserts dma_finish in order to signal block completion. This uses the same timing as dma_ack and forms a handshaking loop with dma_req.
dma_last	Input	Last transaction in block. When the peripheral is the flow controller, it asserts dma_last on the same cycle as dma_req is asserted in order to signal that this transaction request is the last in the block; the block transfer is complete after this transaction is complete. If dma_single is high in the same cycle, then the last transaction is a single transaction. If dma_single is low in the same cycle, then the last transaction is a burst transaction.
dma_req	Input	Transaction request from peripheral. An active level on dma_req initiates a transaction request. The type of transaction – single or burst – is qualified by dma_single. Once dma_req is asserted, it must remain asserted until dma_ack is asserted. When the peripheral that is driving dma_req determines that dma_ack is asserted, it must de-assert dma_req.
dma_single	Input	Single or burst transaction request. If dma_single is de-asserted in the same clock cycle as a rising edge on dma_req, a burst transaction is requested by the peripheral. If asserted, the peripheral requests a single transaction.

The following timing diagrams assume that handshaking interface 0 is active-high.

Fig 9-15 shows a burst transaction followed by a single transaction, where the single transaction is the last in the block. On clock edge T1, DMAC samples that dma_req[0] is asserted, dma_single[0] is de-asserted, and dma_last[0] is de-asserted. This is a request for a burst transaction, which is not the last transaction in the block.

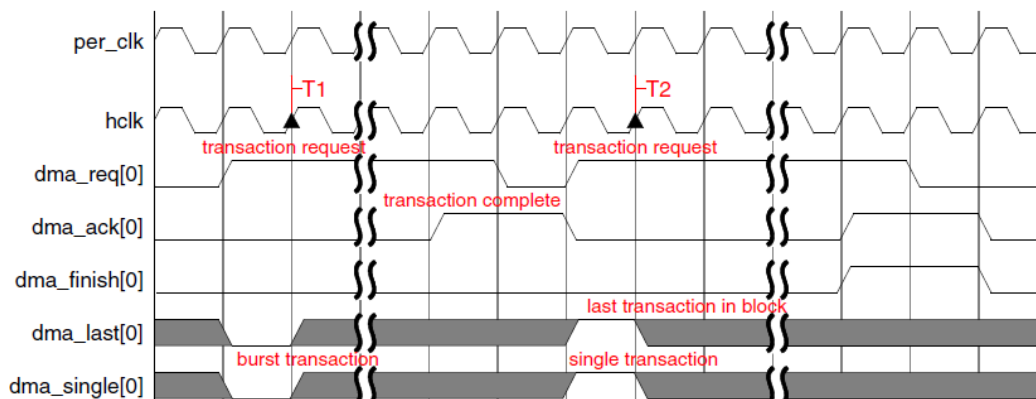


Fig 9-15 Burst transaction followed by single transaction that terminates block

On clock edge T2, the DMAC samples that `dma_req[0]`, `dma_single[0]`, and `dma_last[0]` are all asserted. This is a request for a single transaction, which is the last transaction in the block. The `dma_last[0]` and `dma_single[0]` signals need only be valid on the same clock cycle that `dma_req` is generated.

Similarly, Fig 9-16 shows a single transaction followed by a burst transaction, where the burst transaction is the last transaction in the block.

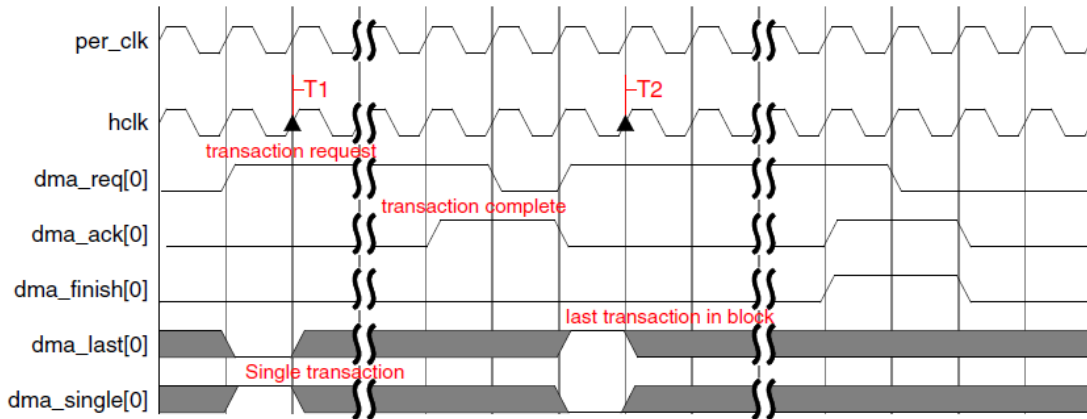


Fig 9-16 Single transaction followed by burst transaction that terminates block

9.2.7.2 Single Transactions – Peripheral is Flow Controller

When the source peripheral is the flow controller, then it can hardcode `dma_single` to an inactive level (hardware handshaking). This occurs when:

$$\text{block_size_bytes_src/src_burst_size_bytes} = \text{integer} \quad (10)$$

When the destination peripheral is the flow controller, then the destination peripheral can hardcode `dma_single` to an inactive level (hardware handshaking). This occurs when:

$$\text{block_size_bytes_dst/dst_burst_size_bytes} = \text{integer} \quad (11)$$

9.2.8 Setting up Transfers

Transfers are set up by programming fields of the CTLx and CFGx registers for that channel. As shown in Fig 9-4, a single block is made up of numerous transactions – single and burst – which are in turn composed of AHB transfers. A peripheral requests a transaction through the handshaking interface to the DMAC. The operation of the handshaking interface is different and depends on what is acting as the flow controller.

Table 9-4 lists the parameters that are investigated in the following examples. The effects of these parameters on the flow of the block transfer are highlighted. In addition to the software parameters, it includes the channel FIFO depth, DMAH_CHx_FIFO_DEPTH, which is configurable only in coreConsultant.

Table 9-4 Parameters Used in Transfer Examples

Parameter	Description
DMAH_CHx_FIFO_DEPTH	Channel x FIFO depth in bytes
CTLx.TT_FC	Transfer type and flow control
CTLx.BLOCK_TS	Block transfer size
CTLx.SRC_TR_WIDTH	Source transfer width
CTLx.DST_TR_WIDTH	Destination transfer width
CTLx.SRC_MSIZ	Source burst transaction length
CTLx.DEST_MSIZ	Destination burst transaction length
CFGx.MAX_ABRST	Maximum AMBA burst length
CFGx.FIFO_MODE	FIFO mode select
CFGx.FCMODE	Flow-control mode

9.2.8.1 Transfer Operation

The following examples show the effect of different settings of each parameter from Table 9-4 on a DMA block transfer. In all examples, it is assumed that no bursts are early-terminated by the AHB system arbiter, unless otherwise stated. Example 1 through Example 8 use hardware handshaking on both the source and destination side.

The following is a brief description of each of the examples:

- Example 1 – Block transfer when the DMAC is the flow controller.
- Example 2 – Effect of DMAH_CHx_FIFO_DEPTH on a block transfer.
- Example 3 – Effect of maximum AMBA burst length, CFGx.MAX_ABRST, on a block transfer.
- Example 4 – Block transfer when the DMAC is the flow controller and the source peripheral enters the Single Transaction Region.
- Example 5 – Block transfer when the DMAC is the flow controller and the destination peripheral enters the Single Transaction Region. Also demonstrates channel FIFO flushing to the destination peripheral at the end of a block transfer.
- Example 6 – Effect of CFGx.FIFO_MODE on a block transfer.
- Example 7 – Block transfer when the destination peripheral is the flow controller and data pre-fetching from the source is enabled; CFGx.FCMODE = 0.
- Example 8 – Block transfer when the destination peripheral is the flow controller and data pre-fetching from the source is disabled; CFGx.FCMODE = 1.

The DMAC is programmed with the number of data items that are to be transferred for each burst transaction request, CTLx.SRC_MSIZ/CTLx.DEST_MSIZ. Similarly, the width of each data item in the transaction is set by the CTLx.SRC_TR_WIDTH and CTLx.DST_TR_WIDTH fields.

9.2.8.1.1 Example 1

Scenario: Example block transfer when the DMAC is the flow controller. Table 9-5 lists the DMA parameters for this example.

Table 9-5 Parameters in transfer operation – Example 1

Parameter	Description
CTLx.TT_FC = 3'b011	Peripheral-to-peripheral transfer with DMAC as flow controller
CTLx.BLOCK_TS = 12	–
CTLx.SRC_TR_WIDTH = 3'b010	32 bits
CTLx.DST_TR_WIDTH = 3'b010	32 bits
CTLx.SRC_MSIZ = 3'b001	Source burst transaction length = 4
CTLx.DEST_MSIZ = 3'b001	Destination burst transaction length = 4
CFGx.MAX_ABRST = 1'b0	No limit on maximum AMBA burst length
DMAH_CHx_FIFO_DEPTH = 16 bytes	–

Using equation (5), a total of 48 bytes are transferred in the block; that is, $blk_size_bytes_dma = 48$. As shown in Fig 9-17, this block transfer consists of three bursts of length 4 from the source, interleaved with three bursts, again of length 4, to the destination.

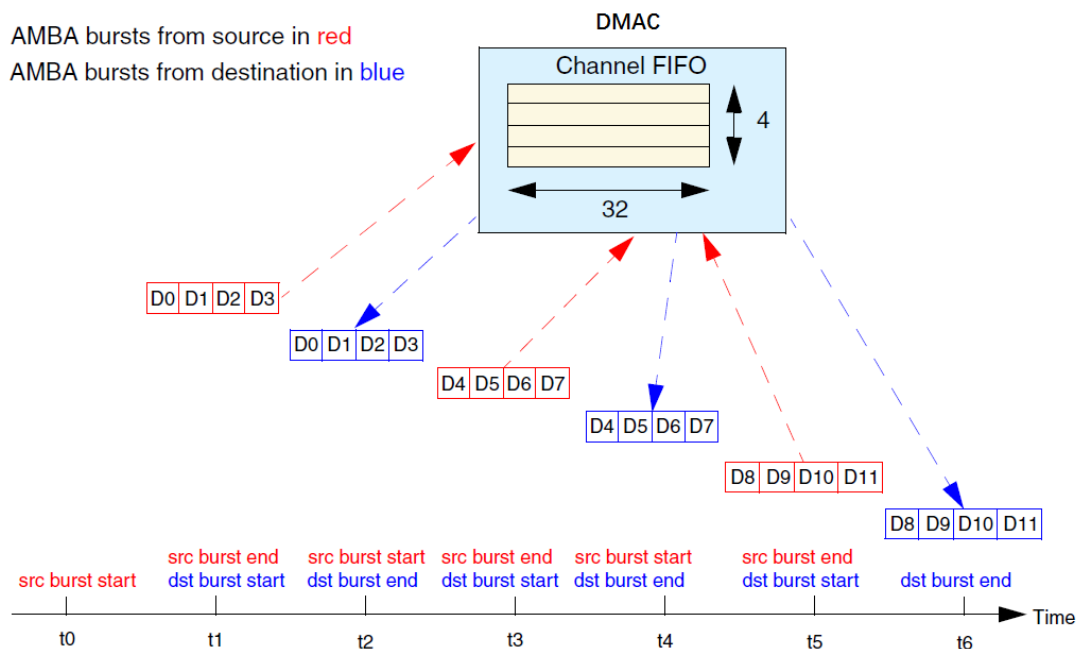


Fig 9-17 Breakdown of Block Transfer

The channel FIFO is alternatively filled by a burst from the source and emptied by a burst to the destination until the block transfer has completed, as shown in Fig 9-18.

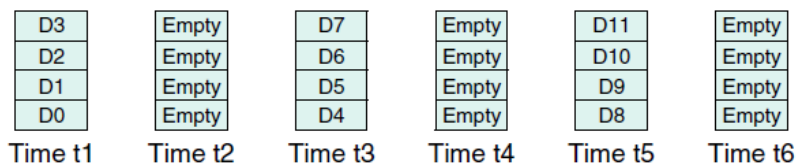


Fig 9-18 Channel FIFO contents at times indicated in Fig 9-17

Burst transactions are completed in one burst. Additionally, because (8) and (9) are both true, neither the source or destination peripherals enter their Single Transaction Region at any stage throughout the DMA transfer, and the block transfer from the source and to the destination consists of burst transactions only.

9.2.8.1.2 Example 2

Scenario: Effect of DMAH_CHx_FIFO_DEPTH on block transfers.

In this example, the coreConsultant DMAH_CHx_FIFO_DEPTH parameter is changed to 8 bytes, and all other parameters are left unchanged from Example 1, Table 9-5.

Example 1 shows the source and destination burst transactions completing in a single burst. In general, a burst transaction may take multiple bursts to complete. With the DMAH_CHx_FIFO_DEPTH parameter set to 8 bytes instead of 16 bytes, the block transfer would look like that shown in Fig 9-19.

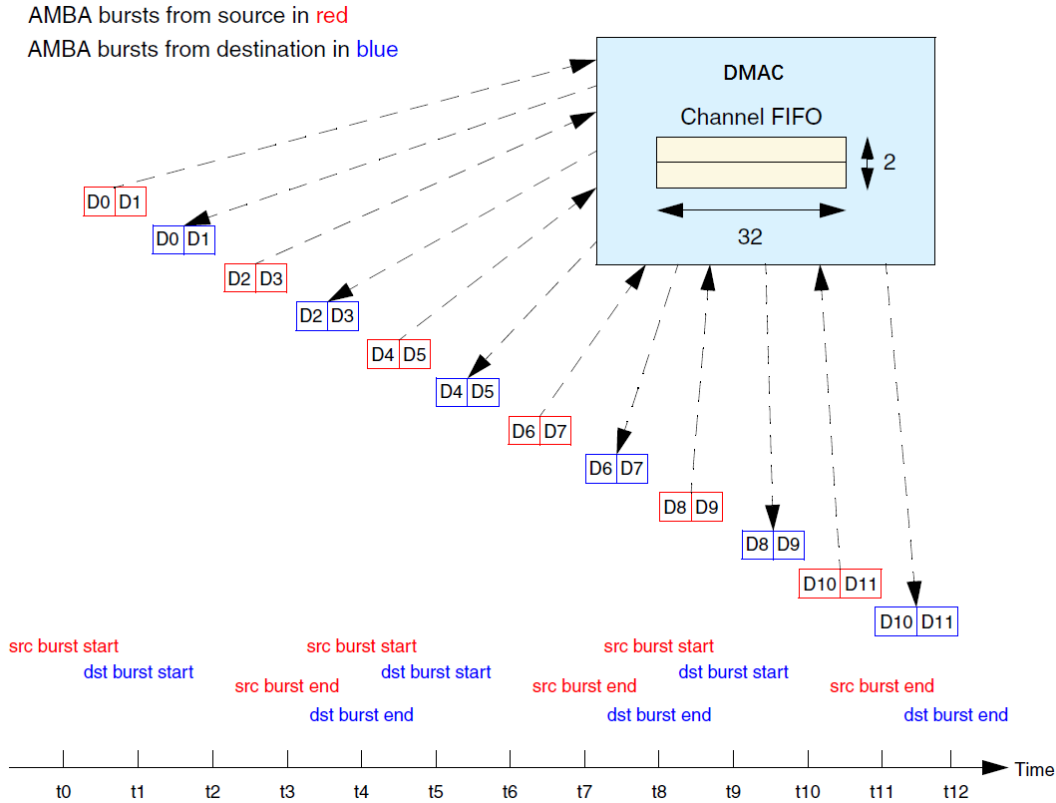


Fig 9-19 Breakdown of block transfer for DMAH_CH_FIFO_DEPTH = 8

The block transfer consists of six bursts of length 2 from the source, interleaved with six bursts – again of length 2 – to the destination, as shown in Fig 9-17. The channel FIFO is alternatively filled by a burst from the source and emptied by a burst to the destination, until the block transfer has completed. In this example, a transfer of each source or destination burst transaction is made up of two bursts, each of length 2.

Therefore, Example 2 has twice the number of bursts per block than Example 1.

Recommendation: To allow a burst transaction to complete in a single burst, the DMAC channel FIFO depth should be large enough to accept an amount of data equal to an entire burst transaction. Therefore, in order to allow both source and destination burst transactions to complete in one burst:

$$\text{DMAH_CHx_FIFO_DEPTH} \geq \max(2 * \text{src_burst_size_bytes}, 2 * \text{dst_burst_size_bytes}) \quad (12)$$

Adhering to the above recommendation results in a reduced number of bursts per block, which in turn results in improved bus utilization and lower latency for block transfers.

9.2.8.1.3 Example 3

Scenario: Effect of the maximum AMBA burst length, CFGx. MAX_ABRST. If the CFGx. MAX_ABRST = 2 parameter and all other parameters are left unchanged from Example 1, Table 9-5, then the block transfer would look like that shown in Fig 9-20.

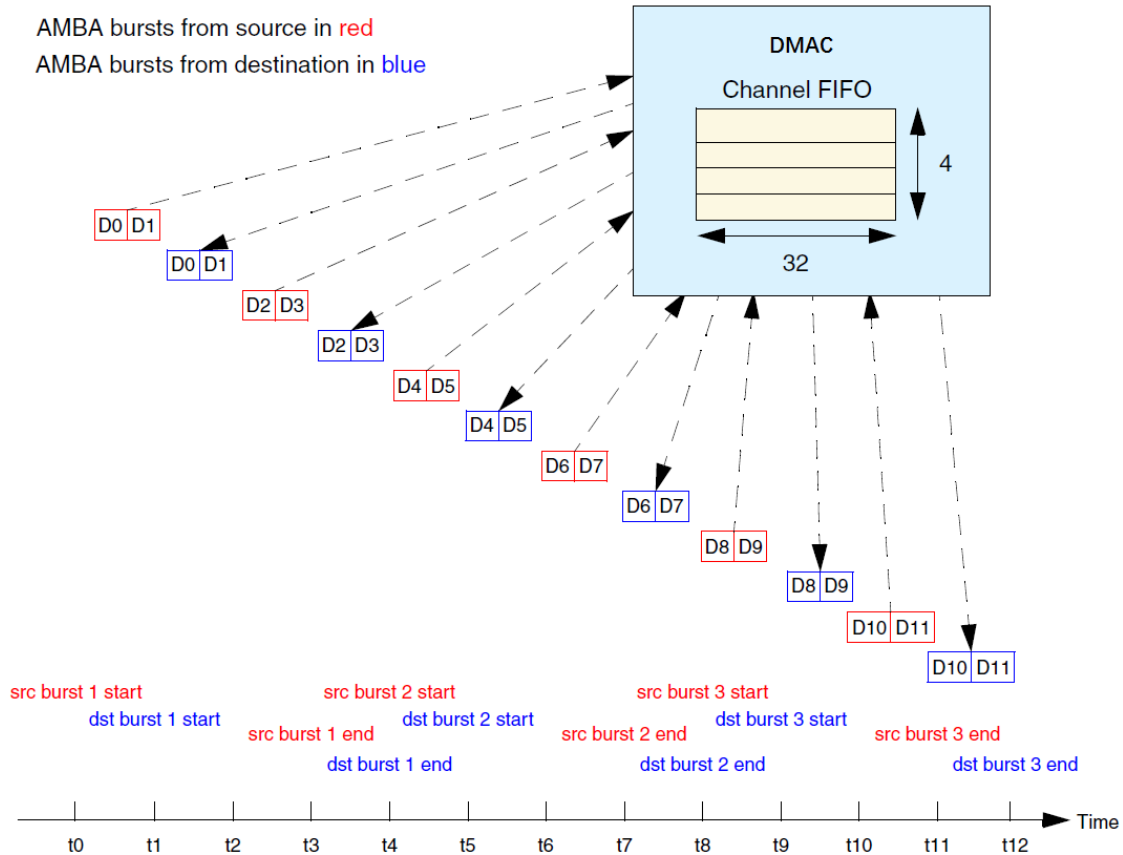


Fig 9-20 Breakdown of block transfer where max_abrst = 2, Case 1

The channel FIFO is alternatively half filled by a burst from the source, and then emptied by a burst to the destination until the block transfer has completed; this is illustrated in Fig 9-21.

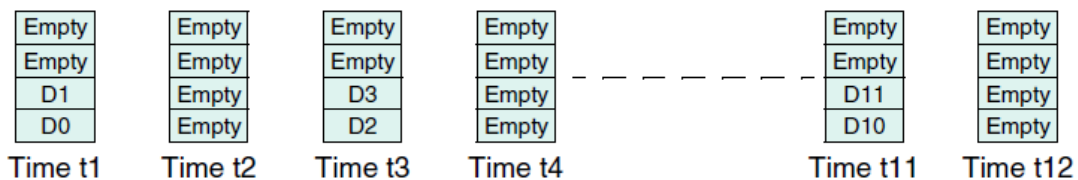


Fig 9-21 Channel FIFO contents at times indicated in Fig 9-20

In this example block transfer, each source or destination burst transaction is made up of two bursts, each of length 2. As Fig 9-21 illustrates, the top two channel FIFO locations are redundant for this block transfer. However, this is not the general case. The block transfer could proceed as indicated in Fig 9-22.

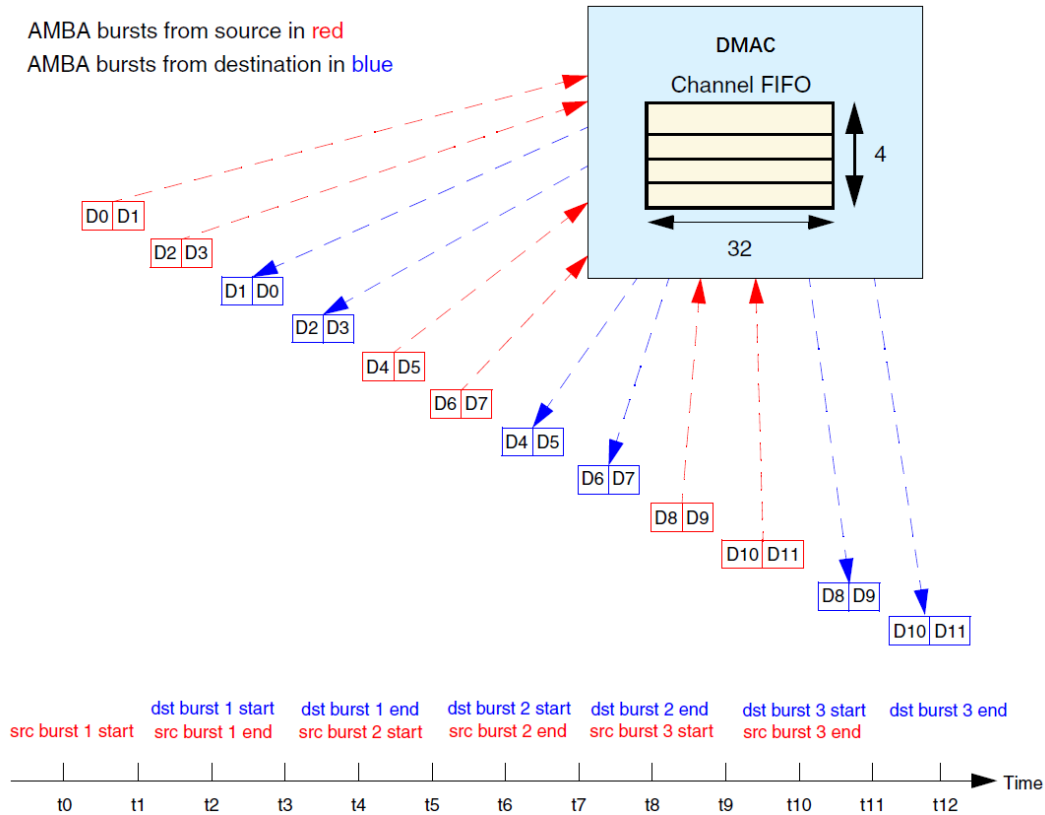


Fig 9-22 Breakdown of block transfer where max_abrst = 2, Case 2

This depends on the timing of the source and destination transaction requests, relative to each other. Fig 9-23 illustrates the channel FIFO status for Fig 9-22.

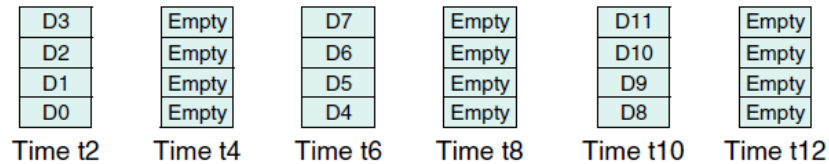


Fig 9-23 Channel FIFO contents at times indicated in Fig 9-22

Recommendation: To allow a burst transaction to complete in a single burst, the following should be true:

$$\text{CFGx.MAX_ABRST} \geq \max(\text{src_burst_size_bytes}, \text{dst_burst_size_bytes})$$

Adhering to the above recommendation results in a reduced number of bursts per block, which in turn results in improved bus utilization and lower latency for block transfers. Limiting a burst to a maximum length prevents the DMAC from saturating the AHB bus when the AHB system arbiter is configured to only allow changing of the grant signals to bus masters at the end of an undefined length burst. It also prevents a channel from saturating a DMAC master bus interface.

9.2.8.1.4 Example 4

Scenario: Source peripheral enters Single Transaction Region; the DMAC is the flow controller.

This example demonstrates how a block from the source can be completed using a series of single transactions. It also demonstrates how the watermark level that triggers a burst request in the source peripheral can be dynamically adjusted so that the block transfer from the source completes with an Early-Terminated Burst Transaction. Table 9-6 lists the parameters used in this example.

Table 9-6 Parameters in transfer operation – Example 4

Parameter	Description
CTLx.TT_FC = 3'b011	Peripheral-to-peripheral transfer with DMAC as flow controller
CTLx.BLOCK_TS = 12	–
CTLx.SRC_TR_WIDTH = 3'b010	32 bits
CTLx.DST_TR_WIDTH = 3'b010	32 bits
CTLx.SRC_MSIZ = 3'b010	Source burst transaction length = 8
CTLx.DEST_MSIZ = 3'b001	Destination burst transaction length = 4
CFGx.MAX_ABRST = 1'b0	No limit on maximum AMBA burst length
DMAH_CHx_FIFD_DEPTH = 16 bytes	–

In this case, BLOCK_TS is not a multiple of the source burst transaction length, CTLx.SRC_MSIZ, so near the end of a block transfer from the source, the amount of data left to be transferred is less than *src_burst_size_bytes*.

In this example, the block size is a multiple of the destination burst transaction length:

$$blk_size_bytes_dma/dst_burst_size_bytes = 48/16 = integer$$

The destination block is made up of three burst transactions to the destination and does not enter the Single Transaction Region.

The block size is not a multiple of the source burst transaction length:

$$blk_size_bytes_dma/src_burst_size_bytes = 48/32 \neq integer$$

Consider the case where the watermark level that triggers a source burst request in the source peripheral is equal to CTLx.SRC_MSIZ = 8; that is, eight entries or more need to be in the source peripheral FIFO in order to trigger a burst request. Fig 9-24 shows how this block transfer is broken into burst and single transactions, and bursts and single transfers.

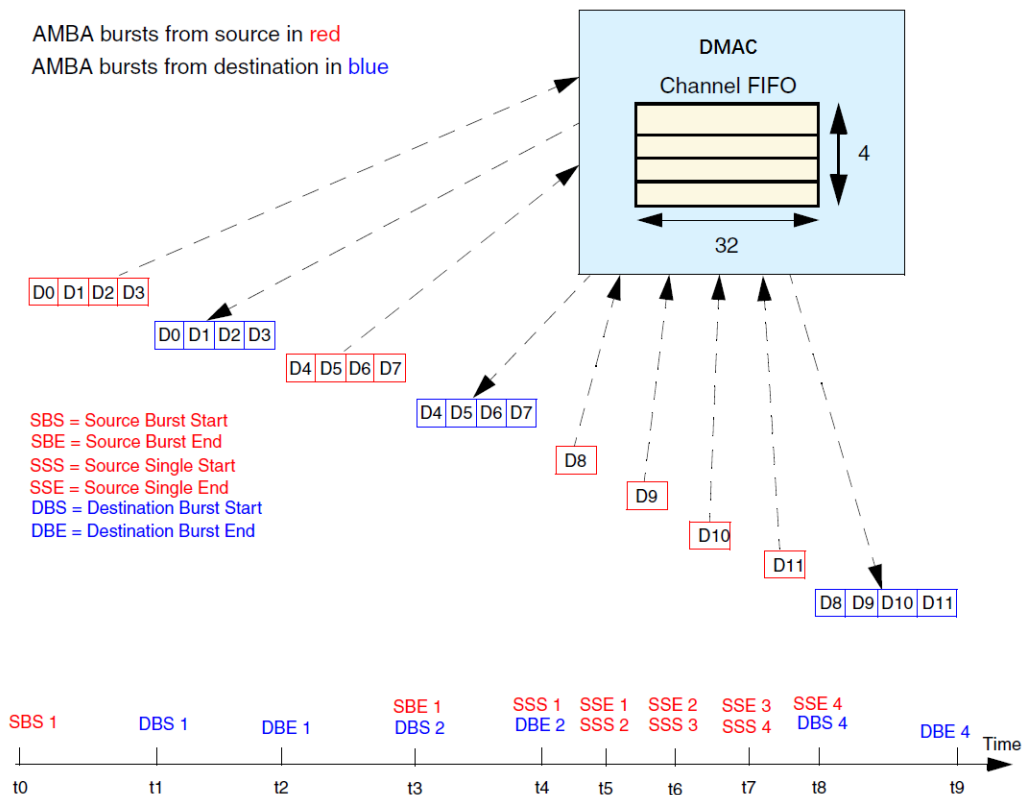


Fig 9-24 Breakdown of block transfer

Fig 9-25 shows the status of the source FIFO at various times throughout the source block transfer.

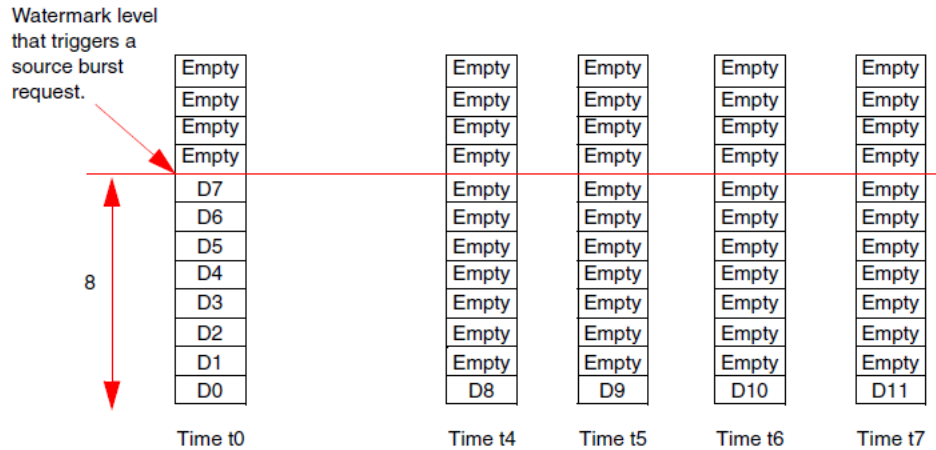


Fig 9-25 Source FIFO contents at time indicated in Fig 9-24

As shown in Fig 9-26, if the DMAC does not perform single transactions, the source FIFO contains four entries at time t1. However, the source has no more data to send. Therefore, if the watermark level remains at 8 (at time t1, Case A in Fig 9-26), the watermark level is never reached and a new burst request is never triggered.

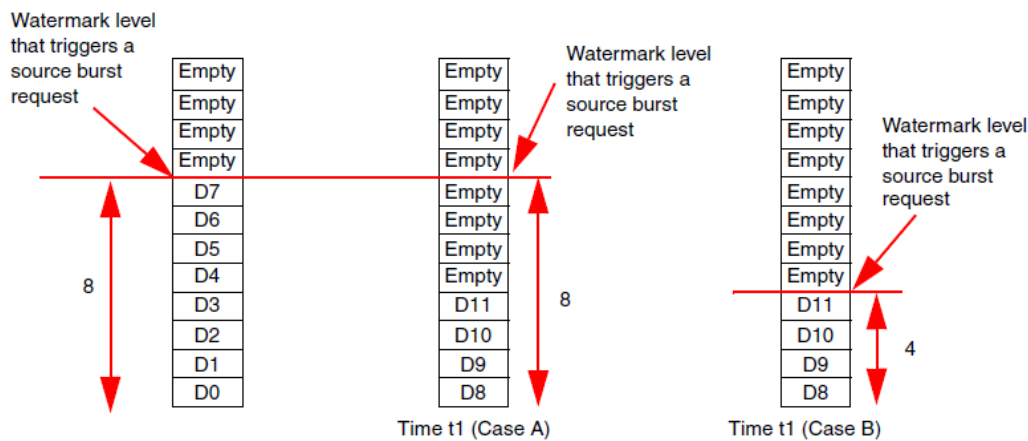


Fig 9-26 Source FIFO contents where watermark level is dynamically adjusted

The source peripheral, not knowing the length of a block and only able to request burst transactions, sits and waits for the FIFO level to reach a watermark level before requesting a new burst transaction request. This region, where the amount of data left to transfer in the source block is less than `src_burst_size_bytes`, is known as the Single Transaction Region.

In the Single Transaction Region, the DMAC performs single transactions from the source peripheral until the source block transfer has completed. In this example, the DMAC completes the source block transfer using four single transactions from the source.

Now consider Case B in Fig 9-26, where the source peripheral can dynamically adjust the watermark level that triggers a burst transaction request near the end of a block. After the first source burst transaction completes, the source peripheral recognizes that it has only four data items left to complete in the block and adjusts the FIFO watermark level that triggers a burst transaction to 4. This triggers a burst request, and the block completes using a burst transaction. However, `CTLx.SRC_MSIZ = 8`, and there are only four data items left to transfer in the source block. The DMAC terminates the last source burst transaction early and fetches only four of the eight data items in the last source burst transaction. This is called an Early-Terminated Burst Transaction.

Observation: Under certain conditions, it is possible to hardcode `dma_single` from the source peripheral to an inactive level (hardware handshaking). Under the same conditions, it is possible for software to complete a source block transfer without initiating single transactions from the source.

9.2.8.1.5 Example 5

Scenario: The destination peripheral enters the Single Transaction Region while the DMAC is the flow controller. This example also demonstrates how the DMAC channel FIFO is flushed at the end of a block transfer to the destination. Consider the case with the parameters set to values listed in Table 9-7.

Table 9-7 Parameters in transfer operation – Example 5

Parameter	Description
CTLx.TT_FC = 3'b011	Peripheral-to-peripheral transfer with DMAC as flow controller
CTLx.BLOCK_TS = 44	–
CTLx.SRC_TR_WIDTH = 3'b000	8 bits
CTLx.DST_TR_WIDTH = 3'b011	64 bits
CTLx.SRC_MSIZ = 3'b001	Source burst transaction length = 4
CTLx.DEST_MSIZ = 3'b001	Destination burst transaction length = 4
CFGx.MAX_ABRST = 1'b0	No limit on maximum AMBA burst length
DMAH_CHx_FIFO_DEPTH = 32 bytes	–

In this example, the block size is a multiple of the source burst transaction length:

$$blk_size_bytes_dma/src_burst_size_bytes = (44 * 1)/4 = 11 = integer$$

The source block transfer is completed using only burst transactions, and the source does not enter the Single Transaction Region.

The block size is not a multiple of the destination burst transaction length:

$$blk_size_bytes_dma/dst_burst_size_bytes = 44/32 \neq integer$$

So near the end of the block transfer to the destination, the amount of data left to be transferred is less than *dst_burst_size_bytes* and the destination enters the Single Transaction Region.

Fig 9-27 shows one way in which the block transfer to the destination can occur.

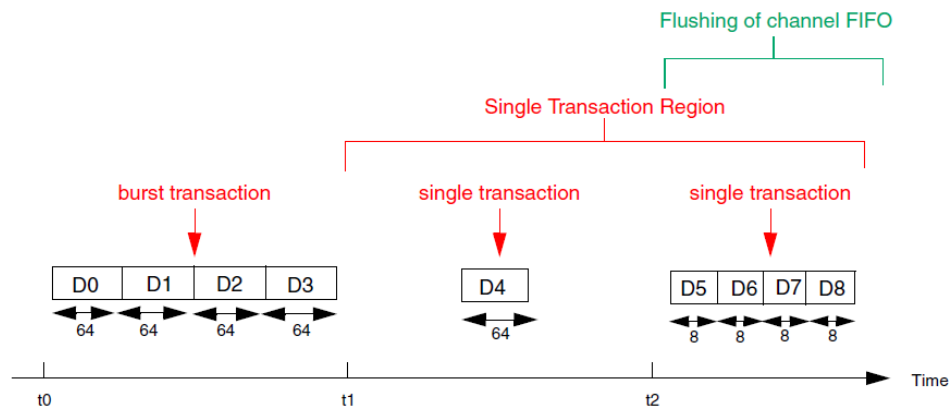


Fig 9-27 Block transfer to destination

After the first 32 bytes (*dst_burst_size_bytes* = 32) of the destination burst transaction have been transferred to the destination, there are 12 bytes (*blk_size_bytes_dma* - *dst_burst_size_bytes* = 44 - 32) left to transfer. This is less than the amount of data that is transferred in a destination burst (*dst_burst_size_bytes* = 32). Therefore, the destination peripheral enters the Single Transaction Region where the DMAC can complete a block transfer to the destination using single transactions.

Note:

- In the Single Transaction Region, asserting *dma_single* initiates a single transaction for hardware handshaking.
- The destination peripheral, not knowing the length of a block and only able to request burst transactions, sits and waits for the FIFO to fall below a watermark level before requesting a new burst transaction request.

At time t2 in Fig 9-27, a single transaction to the destination has been completed. There are now only four bytes (12 - *dst_single_size_bytes* = 12 - 8) left to transfer in the destination block. However, CTLx.DST_TR_WIDTH implies 64-bit AHB transfers to the destination (*dst_single_size_bytes* = 8 byte); therefore, the DMAC cannot form a single word of the specified CTLx.DST_TR_WIDTH.

The DMAC channel FIFO has four bytes in it that must be flushed to the destination. The DMAC switches into a “FIFO flush mode,” where the block transfer to the destination is completed by changing the AHB transfer width to the destination to be equal to that of the CTLx.SRC_TR_WIDTH; that is, byte AHB transfers in this example. Thus the last single transaction in the destination block is made up of a burst of length 4 and CTLx.SRC_TR_WIDTH width.

When the DMAC is in FIFO flush mode, the address on haddr is incremented by the value of hsize on the bus; that is, CTLx.SRC_TR_WIDTH, and not CTLx.DST_TR_WIDTH. In cases where the DARx is selected to be contiguous between blocks, the DARx will need re-alignment at the start of the next block, since it is aligned to CTLx.SRC_TR_WIDTH and not CTLx.DST_TR_WIDTH at the end of the previous block. This is handled by hardware.

In general, channel FIFO flushing to the destination occurs if all three of the following are true:

- DMAC or the Source peripheral are flow control peripherals
- CTLx.DST_TR_WIDTH > CTLx.SRC_TR_WIDTH
- Flow control device:
 - If DMAC is flow controller:
 $blk_size_bytes_dma/dst_single_size_bytes \neq integer$
 - If source is flow controller:
 $blk_size_bytes_src/dst_single_size_bytes \neq integer$

Note: When not in FIFO flush mode, a single transaction is mapped to a single AHB transfer. However, in FIFO flush mode, a single transaction is mapped to multiple AHB transfers of CTLx.SRC_TR_WIDTH width. The cumulative total of data transferred to the destination in FIFO flush mode is less than *dst_single_size_bytes*.

In the above example, a burst request is not generated in the Single Transaction Region. If a burst request were generated at time t1 in Fig 9-27, then the burst transaction would proceed until there was not enough data left in the destination block to form a single data item of CTLx.DST_TR_WIDTH width. The burst transaction would then be early-terminated. In this example, only one data item of the four requested (decoded value of DEST_MIZE = 4) would be transferred to the destination in the burst transaction. This is referred to as an Early-Terminated Burst Transaction. If a burst request were generated at time t2 in Fig 9-27, then the destination block would be completed (four byte transfers to the destination to flush the DMAC channel FIFO) and this burst request would again be early-terminated at the end of the destination block.

Observation: If the source transfer width – CTLx.SRC_TR_WIDTH in the channel control register (CTLx) – is less than the destination transfer width (CTLx.DST_TR_WIDTH), then the FIFO may need to be flushed at the end of the block transfer. This is done by setting the AHB transfer width of the last few AHB transfers of the block to the destination so that it is equal to CTLx.SRC_TR_WIDTH and not the programmed CTLx.DST_TR_WIDTH.

9.2.8.1.6 Example 6

Scenario: In all examples presented so far, none of the bursts have been early-terminated by the AHB system arbiter. Referring to Example 1, the AHB transfers on the source and destination side look somewhat symmetric. In the examples presented so far, where the bursts are not early-terminated by the AHB system arbiter, the traffic profile on the AHB bus would be the same, regardless of the value of CFGx.FIFO_MODE. This example, however, considers the effect of CFGx.FIFO_MODE.

CFGx.FIFO_MODE: Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced. 0 = Space/data available for single AHB transfer of the specified transfer width. 1 = Data available is greater than or equal to half the FIFO depth for destination transfers and space available is greater than half the fifo depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.

Table 9-8 lists the parameters used in this example.

Table 9-8 Parameters in transfer operation – Example 6

Parameter	Description
CTLx.TT_FC = 3'b011	Peripheral-to-peripheral transfer with DMAC as flow controller
CTLx.BLOCK_TS = 32	–
CTLx.SRC_TR_WIDTH = 3'b010	32 bits
CTLx.DST_TR_WIDTH = 3'b010	32 bits
CTLx.SRC_MSIZ = 3'b010	Decode value = 8
CTLx.DEST_MSIZ = 3'b001	Decode value = 4

CFGx.MAX_ABRST = 1'b0	No limit on maximum AMBA burst length
DMAH.CHx_FIFO_DEPTH = 16 bytes	–

The block transfer may proceed by alternately filling and emptying the DMAC channel FIFO. Up to time t4, the transfer might proceed like that shown in Fig 9-28.

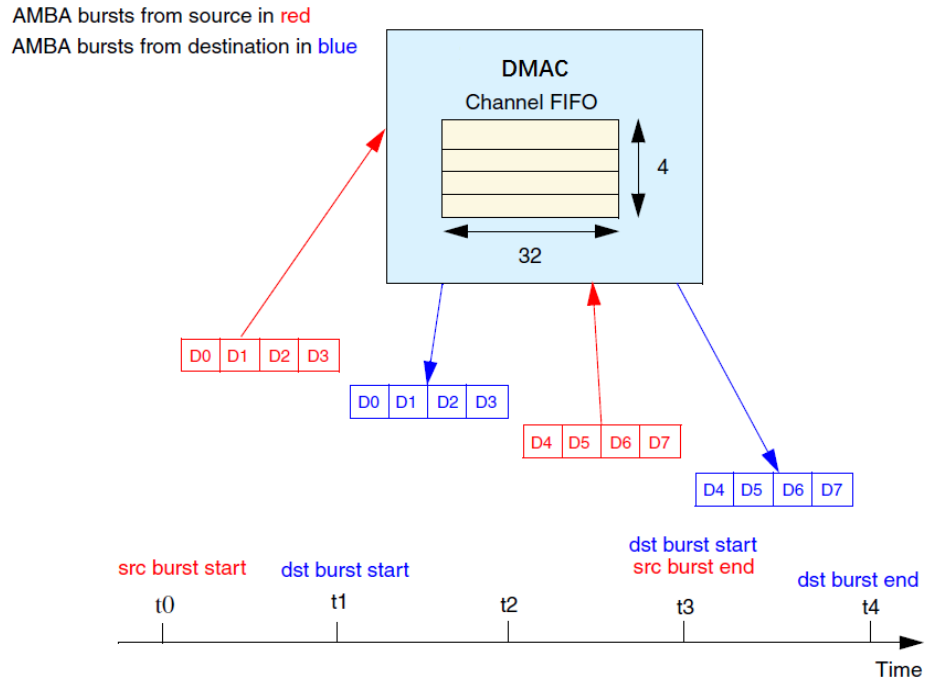


Fig 9-28 Block transfer up to time 't4'

At time t4, the src, channel, and destination FIFOs might look like that shown in Fig 9-29.

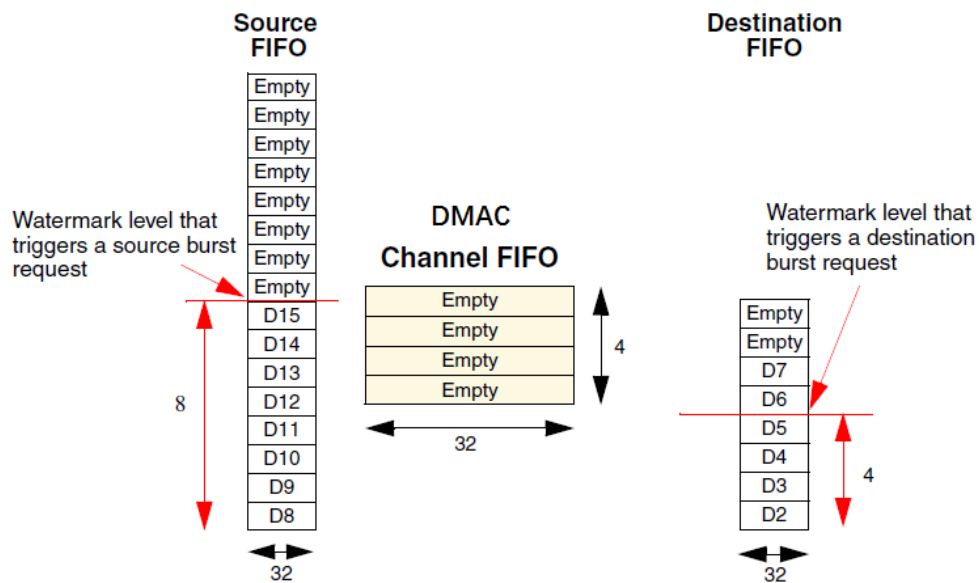


Fig 9-29 Source, DMAC channel and destination FIFOs at time 't4' in Fig 9-26

At time 't4', a source burst transaction is requested, and the DMAC attempts a burst of length 4. Suppose that this burst is early-burst terminated after three AHB transfers. The FIFO status after this burst might look like that shown in Fig 9-30.

Referring to Fig 9-30, notice that a burst request from the destination is not triggered, since the destination FIFO contents are above the watermark level. The DMAC has space for one data item in the channel FIFO. So, does the DMAC initiate a single AHB transfer from the source peripheral to fill the channel FIFO?

The answer depends on the value of CFGx.FIFO_MODE. If CFGx.FIFO_MODE = 0, then the DMAC attempts to perform a single AHB transfer in order to fill the channel FIFO. If CFGx.FIFO_MODE = 1, then the DMAC waits until the channel FIFO is less than half-full before initiating a burst from the source, as illustrated in Fig 9-30.

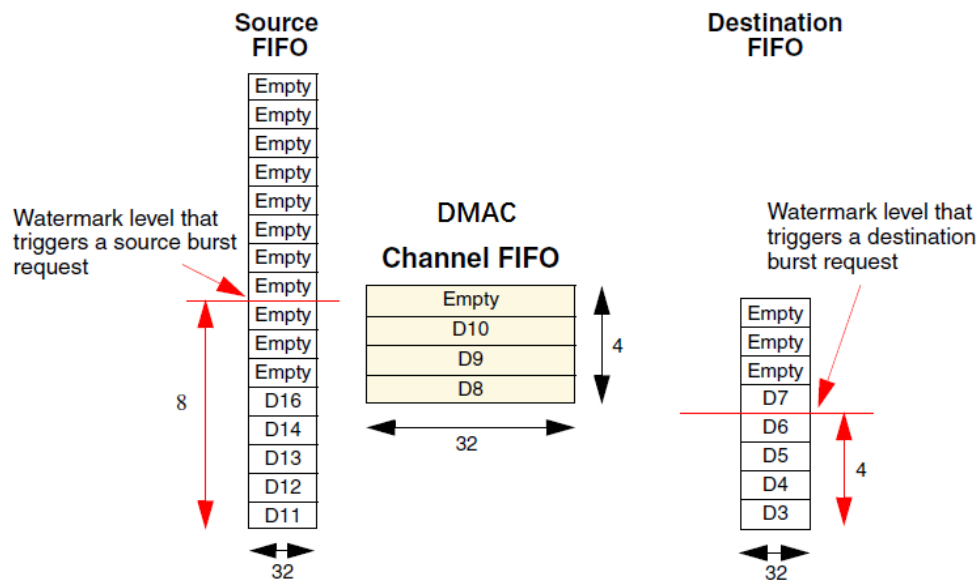


Fig 9-30 FIFO status after early-terminated burst

Observation: When CFGx.FIFO_MODE = 1, the number of bursts per block is less than when CFGx.FIFO_MODE = 0 and, hence, the bus utilization will improve. This setting favors longer bursts. However, the latency of DMA transfers may increase when CFGx.FIFO_MODE = 1, since the DMAC waits for the channel FIFO contents to be less than half the FIFO depth for source transfers, or greater than or equal to half the FIFO depth for destination transfers. Therefore, system bus occupancy and usage can be improved by delaying the servicing of multiple requests until there is sufficient data/space available in the FIFO to generate a burst (rather than multiple single AHB transfers); this comes at the expense of transfer latency. For reduced block transfer latency, set CFGx.FIFO_MODE = 0. For improved bus utilization, set CFGx.FIFO_MODE = 1.

9.2.8.1.7 Example 7

Scenario: Example block transfer when the destination is the flow controller; the effect of data pre-fetching (CFGx.FCMODE = 0) and possible data loss.

Note: Data pre-fetching is when data is fetched from the source before the destination requests it.

Flow Control Mode: Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.

- 0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled.
- 1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited such that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.

Table 9-9 lists the parameters used in this example.

Table 9-9 Parameters in transfer operation – Example 7

Parameter	Description
CTLx.TT_FC = 3'b111	Peripheral-to-peripheral transfer with DMAC as flow controller
CTLx.BLOCK_TS = x	–
CTLx.SRC_TR_WIDTH = 3'b010	32 bits
CTLx.DST_TR_WIDTH = 3'b010	32 bits
CTLx.SRC_MSIZ = 3'b010	Decode value = 8
CTLx.DST_MSIZ = 3'b001	Decode value = 4
CFGx.MAX_ABRST = 1'b0	No limit on maximum AMBA burst length
DMAH_CHx_FIFO_DEPTH = 32 bytes	–
CFGx.FCMODE = 0	Data pre-fetching enabled
CFGx.SRC_PER = 0	Source assigned handshaking interface 0
CFGx.DST_PER = 1	Destination assigned handshaking interface 1
CFGx.MAX_ABRST = 7	–

Consider a case where the destination block is made up of one burst transaction, followed by one single transaction.

$$blk_size_bytes_dst = dst_burst_size_bytes = 16 + 4 = 20 \text{ bytes}$$

There are a number of different cases that can arise when CFGx.FCMODE = 0:

- (1) When the destination peripheral signals a last transaction, there is enough data in the DMAC channel FIFO to complete the last transaction to the destination. Therefore, the DMAC stops transferring data from the source, and the block transfer from the source completes; any surplus data that has been fetched from the source is effectively lost. Two cases arise when the destination peripheral signals the last transaction in a block:
 - a) Active burst transaction request on source side
 - b) No active burst request on source side
- (2) When the destination peripheral signals a last transaction, there is not enough data in the channel FIFO to complete the last transaction to the destination. The DMAC fetches just enough data to complete the block transfer. Two cases arise.
 - a) Source enters Single Transaction Region when destination peripheral signals last transaction.
 - b) Source does not enter Single Transaction Region when destination peripheral signals last transaction.

Setting CFGx.FCMODE is pertinent only when the destination peripheral is the flow controller. When CFGx.FCMODE = 0, scenarios arise where not all the data that has been pre-fetched from the source is required to complete the block transfer to the destination. This excess data is not transferred to the destination peripheral and is effectively lost for a read-sensitive source peripheral. In this example, we assume a read-sensitive source peripheral.

Case a and Case b highlight instances where data pre-fetching is enabled and data is lost. Case a and Case b highlight instances where data pre-fetching is enabled, but no data loss occurs.

In this example, handshaking interface 0 is assigned to the source peripheral, and handshaking interface 1 is assigned to the destination peripheral.

Consider the block transfer shown in Fig 9-31, where the destination is the flow controller and data pre-fetching is enabled (CFGx.FCMODE = 0).

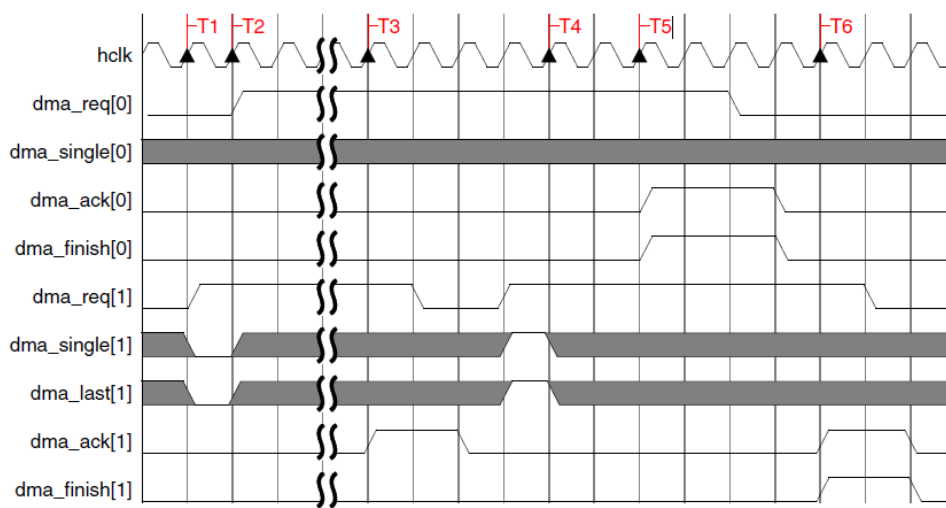


Fig 9-31 Data loss when pre-fetching is enabled

The source requests a burst transaction at time T2. The destination requests a burst transaction at time T1 and completes this burst request at time T3. At time T4, the destination requests a single transaction, which is to be the last in the block transfer. Suppose that at time T4 the DMAC has fetched seven words from the source and written four words to the destination. Therefore, the DMAC has three words in the channel FIFO, but the destination requires only one word to complete the block transfer.

The DMAC, recognizing that it has enough data in the channel FIFO to complete the block transfer to the destination, fetches no more data from the source and early-terminates the source burst transaction (only seven of the eight data items in the source burst transaction have been fetched from the source) – Early-Terminated Burst Transaction. The DMAC asserts `dma_finish[0]` to the source at time T5, and this has the same timing as `dma_ack[0]`, as shown in Fig 9-31.

At time T6, the last single transaction to the destination has completed, which has removed one of the remaining three data words in the channel FIFO. At this time, both the source and destination block transfers have completed, and there remain two data words in the channel FIFO that have been fetched from the source. These two data words are lost because they do not form the start of the next block transfer for multi-block transfers, since the channel FIFO is cleared between blocks for multi-block transfers.

Note: There is an exception to Case a. If the last AHB transfer to the source received a SPLIT/RETRY response over the AHB bus, then `dma_finish` is not asserted until the AHB transfer that received the SPLIT/RETRY response is retried and an OKAY response is received over the hresp AHB bus; to do otherwise would be a violation of the AMBA protocol. This additional word that is fetched is effectively lost, since it is not transferred to the destination.

Case 1b – Timing exception on `dma_finish` to the source when data pre-fetching is enabled

Consider the block transfer shown in Fig 9-32, where the destination is the flow controller and data pre-fetching is enabled (`CFGx.FCMODE = 0`).

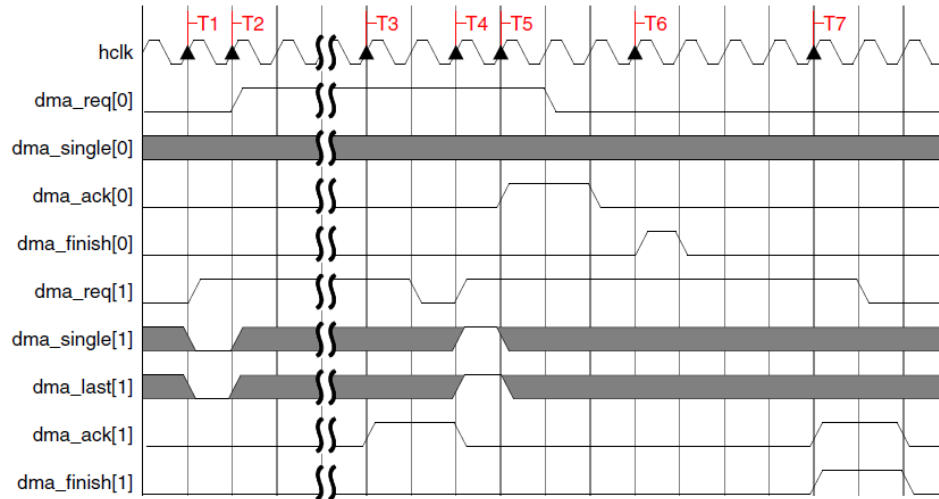


Fig 9-32 Timing exception on dma_finish to source peripheral

The source requests a burst transaction at time T2 and completes the burst transaction at time T5. The destination requests a burst transaction at time T1 and completes this burst request at time T3. At time T4, the destination requests a single transaction, which is to be the last in the block transfer. At time T5, the DMAC has completed the burst transaction from the source.

At time T5, the DMAC has fetched eight words from the source and written four words to the destination, which means that the DMAC has four words in the channel FIFO. However, the destination requires only one word to complete the block transfer. The DMAC, recognizing that it has enough data in the channel FIFO to complete the block transfer to the destination, fetches no more data from the source and signals a source block completion by asserting dma_finish[0] for a single cycle at time T6. Since there is no active transaction on the source side – that is, the previous source burst transaction has completed and there has been no new burst request from the source – the dma_finish[0] cannot form a handshaking loop with dma_req[0] (there is no active burst request) and therefore is asserted for only a single cycle.

Similar to Case a, when both the source and destination block transfers have completed at time T7, there are three data items left in the channel FIFO that are effectively lost.

Case 2a – Data pre-fetching enabled but no data loss. Source enters Single Transaction Region when destination signals last transaction

Consider the block transfer shown in Fig 9-33, where the destination is a flow controller and data pre-fetching is enabled (CFGx.FCMODE = 0). The transfer parameters are the same as case 1a, Table 9-9 Parameters in transfer operation – Example 7.

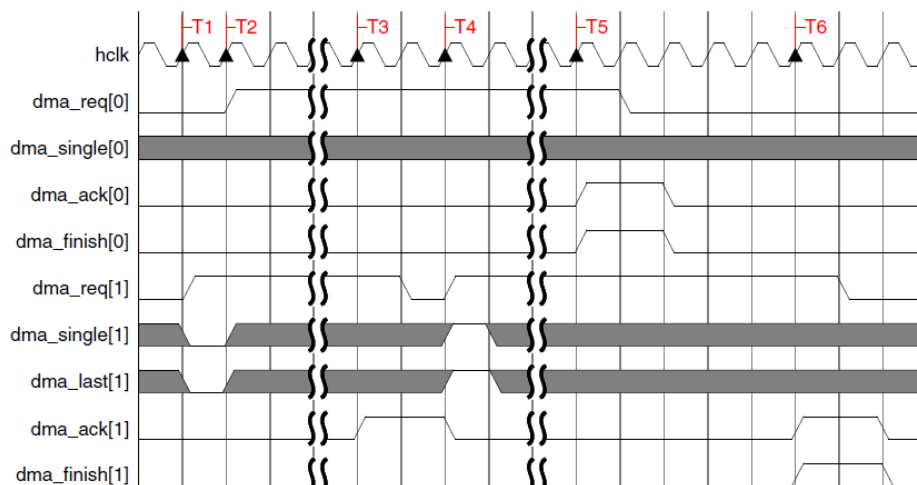


Fig 9-33 Case of no data loss when pre-fetching is enabled

In this scenario, when `dma_last[1]` is asserted by the destination peripheral at time T4, there is not enough data in the channel FIFO to complete the last single transaction. Assume that the DMAC has fetched four data items from the source peripheral at time T4. In this case, the DMAC fetches one more data item from the source peripheral and then early terminates the source burst using an Early-Terminated Burst Transaction. The DMAC signals block completion to the source by asserting `dma_finish[0]` at T5, which forms a handshaking loop with `dma_req[0]`. In this case, there is no data loss, and all data that was fetched from the source has been transferred to the destination.

Consider the case where the transfer parameters are as given in Table 9-10.

Table 9-10 Transfer parameters

Parameter	Description
CTLx.TT_FC = 3'b111	Peripheral-to-peripheral transfer with DMAC as flow controller
CTLx.BLOCK_TS = x	–
CTLx.SRC_TR_WIDTH = 3'b010	32 bits
CTLx.DST_TR_WIDTH = 3'b010	32 bits
CTLx.SRC_MSIZ = 3'b001	Decode value = 4
CTLx.DEST_MSIZ = 3'b001	Decode value = 4
CFGx.MAX_ABRST = 1'b0	No limit on maximum AMBA burst length
DMAH_CHx_FIFO_DEPTH = 32 bytes	–
CFGx.FCMODE = 0	Data pre-fetching enabled
CFGx.SRC_PER = 0	Source assigned handshaking interface 0
CFGx.DEST_PER = 1	Destination assigned handshaking interface 1
CFGx.MAX_ABRST = 7	–

As illustrated in Fig 9-34, the source requests a burst transaction at time T2 and completes the burst transaction at time T3. The destination requests a burst transaction at time T1 and completes this burst request at time T4. The destination requests a last single transaction at time T5; the channel FIFO is empty at this time. The amount of data left to complete a source block transfer – 4 bytes – is less than the following:

$$\text{src_burst_size_bytes} = 4 * 4 = 16 \text{ bytes}$$

Therefore, the source enters the Single Transaction Region at time T6. At time T7, the DMAC samples that `dma_single[0]` from the source peripheral is asserted and initiates a single transaction.

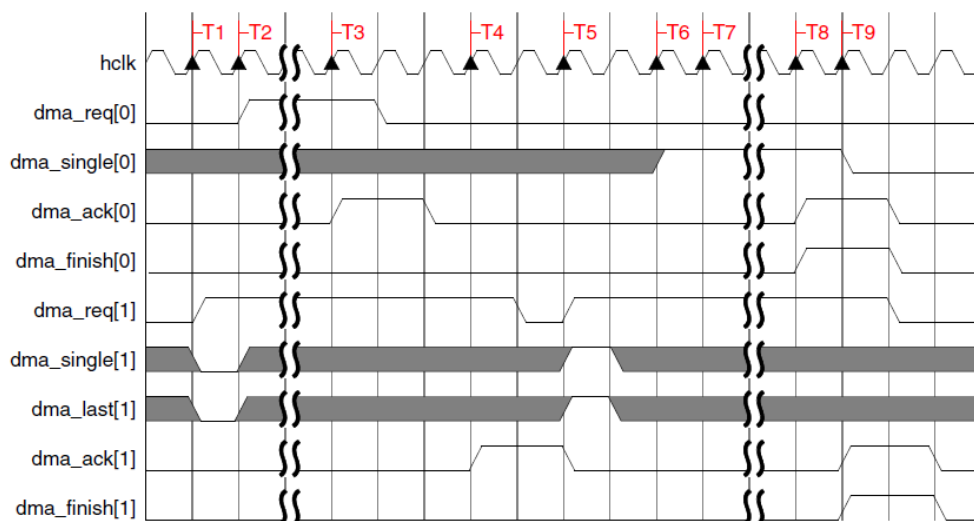


Fig 9-34 Source enters single transaction region when destination asserts `dma_last[1]`

Note: If an active level on `dma_req[0]` was triggered at time T6 or T7, a source burst transaction would have taken precedence over the source single transaction. Upon completion of the source block, this burst transaction would have been early-terminated using an Early-Terminated Burst Transaction.

When this transaction completes at time T8, the DMAC recognizes that enough data has been fetched from the source peripheral to complete the block transfer to the destination. The DMAC asserts `dma_finish[0]` to the source peripheral at time T8; this has the same timing as `dma_ack[0]`. The destination block transfer completes as previously described. No data loss occurs.

Case 2b – Data pre-fetching enabled but no data loss

In this case, the source does not enter the Single Transaction Region when the destination signals the last transaction. This case uses the parameters listed in Table 9-10.

The destination block is made up of two burst transactions.

$$blk_size_bytes_dst = 2 * (4 * 4) = 32 \text{ bytes}$$

As illustrated in Fig 9-35, the source requests a burst transaction at time T2 and completes the burst transaction at time T3. The destination requests a burst transaction at time T1 and completes this burst request at time T4.

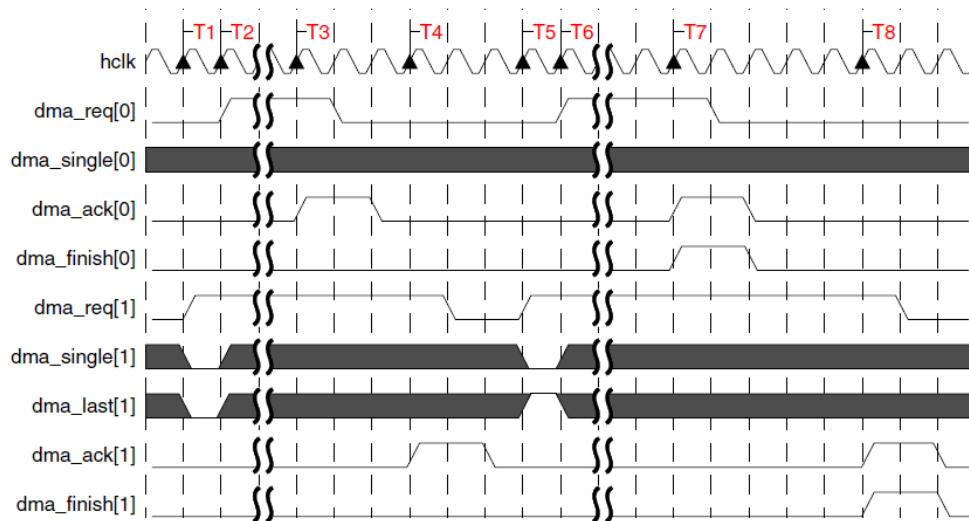


Fig 9-35 Case where source does not enter single transaction region when destination asserts `dma_last[1]`

At time T5 the destination peripheral requests the last burst transaction in the block transfer to the destination. At this point, the channel FIFO is empty. The number of bytes that must be fetched from the source peripheral to complete the block transfer to the destination is equal to $4 * 4 = 16$ bytes. Since 16 bytes is not less than `src_msize_bytes` (16 bytes), the source does not enter the Single Transaction Region. The DMAC waits for a burst request from the source peripheral, which occurs at time T6. Upon completion of this burst request at time T7, the DMAC signals a source block transfer completion and asserts `dma_finish[0]`. Upon completion of the last destination burst transaction at time T8, the DMAC signals a destination block transfer completion and asserts `dma_finish[1]` to the destination.

Note that when data pre-fetching is enabled, `CFGx.FCMODE = 0`, the maximum amount of data that can be lost depends on whether the last transaction in the block transfer to the destination is a single transaction or burst transaction. In the worst case scenario, the DMAC has pre-fetched enough data from the source to fill the channel FIFO when the last transaction is signalled by the destination peripheral.

The maximum amount of data that can be lost is:

- Last transaction in block transfer is a single transaction:
 $DMAH_CHx_FIFO_DEPTH - dst_single_size_bytes$ [refer to equation (1)]
- Last transaction in block transfer is a burst transaction:
 $DMAH_CHx_FIFO_DEPTH - dst_burst_size_bytes$ [refer to equation (2)]
 If this equation is ≤ 0 , then no data is lost.

Thus, if the last transaction in the block is a burst transaction and equation (2) is less than zero, then no data can be lost when `CFGx.FCMODE = 0`. There is one exception to this, as outlined in Example 8.

Enabling data pre-fetching may reduce the latency of the DMA transfer when the destination is the flow controller.

Observation: For a source peripheral that is not read-sensitive (such as memory), data pre-fetching should be enabled – that is, CFGx.FCMODE = 0 – in order to reduce the transfer latency when the destination is the flow controller. If the source peripheral is a read-sensitive device (such as a source FIFO), then data pre-fetching should be disabled – that is, CFGx.FCMODE = 1 – when the destination peripheral is the flow controller.

9.2.8.1.8 Example 8

Scenario: Data loss when destination is flow controller and data pre-fetching is disabled; CFGx.FCMODE = 1.

This scenario arises when CFGx.FCMODE = 1:

- CTLx.SRC_TR_WIDTH ≤ CTLx.DST_TR_WIDTH
- CTLx.SRC_TR_WIDTH > CTLx.DST_TR_WIDTH

Case 1 – CTLx.SRC_TR_WIDTH ≤ CTLx.DST_TR_WIDTH

In this case, the DMAC controls the transfer of data from the source, such that at any time there is at most enough data to complete the current transaction – single or burst – to the destination. If there is currently no active transaction to the destination, then the channel FIFO is empty and no data is pre-fetched from the source, even if the source has an active transaction request. If both the source and destination are requesting, then the DMAC fetches only enough data from the source to complete the current destination transaction, and no more. Therefore, there can never be any data loss.

Case 2 – CTLx.SRC_TR_WIDTH > CTLx.DST_TR_WIDTH

In this example, assume the parameters in Table 9-11.

Table 9-11 Parameters in transfer operation – Example 7, Case 2b

Parameter	Description
CFGx.FCMODE = 1	Data pre-fetching disabled
CTLx.BLOCK_TS = x	–
CTLx.SRC_MSIZ = 3'b001	Decode value = 4
CTLx.DEST_MSIZ = 3'b001	Decode value = 4
CTLx.SRC_TR_WIDTH = 3'b011	64 bits
CTLx.DST_TR_WIDTH = 3'b010	32 bits
CTLx.TT_FC = 3'b111	Peripheral to Peripheral transfer with destination as flow controller
DMAH_CHx_FIFO_DEPTH = 32 bytes	–
CFGx.SRC_PER = 0	Source assigned handshaking interface 0
CFGx.DEST_PER = 1	Destination assigned handshaking interface 1
CFGx.MAX_ABRST = 8	–

Consider the case where the destination block is made up of a burst transaction, followed by one single transaction:

$$\begin{aligned}
 blk_size_bytes_dst &= (8 * 4) + 4 = 36 \text{ bytes} \\
 src_single_size_bytes &= 8 \\
 dst_single_size_bytes &= 4 \\
 src_burst_size_bytes &= 4 * 8 = 32 \\
 dst_burst_size_bytes &= 8 * 4 = 32
 \end{aligned}$$

As illustrated in Fig 9-36, the source requests a burst transaction at time T2, and completes the burst transaction at time T3. The destination requests a burst transaction at time T1 and completes this burst request at time T4. The destination requests a last single transaction at time T5. The channel FIFO is empty at this time.

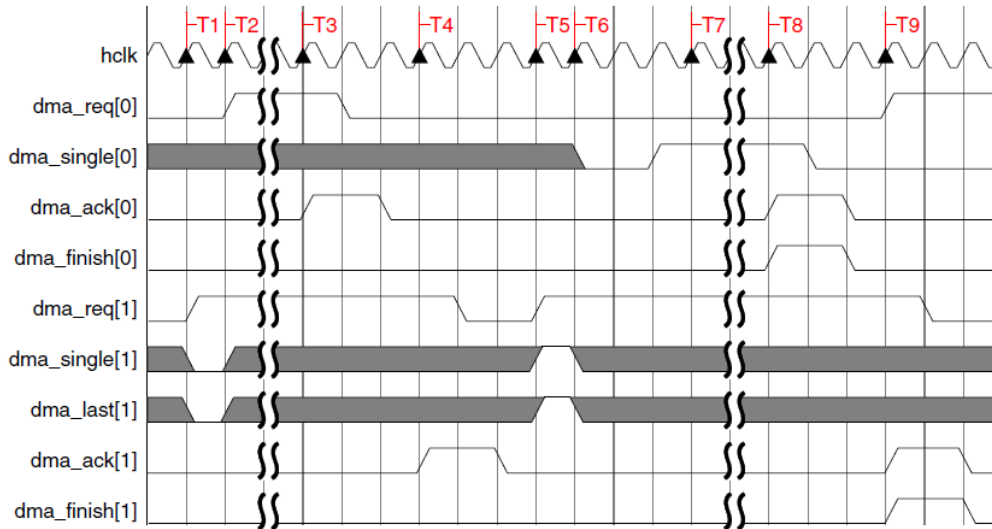


Fig 9-36 Data loss when data pre-fetching is disabled

The amount of data left to complete a source block transfer, 4 bytes, is less than *src_burst_size_bytes* (32 bytes). Therefore, the source enters the Single Transaction Region. At time T7, the DMAC samples that *dma_single[0]* is asserted and initiates a single transaction from the source.

The DMAC fetches a single source data item from the source peripheral and stores the eight bytes in the channel FIFO. This single transaction completes at time T8; the source block also completes at time T8. However, the destination requires only four of these eight bytes to complete the block transfer to the destination (the block transfer to destination completes at time T9). The remaining four bytes are lost. Thus, when the destination is the flow controller, data loss occurs when *CFGx.FCMODE* = 1 if both of the following are true:

- *SRC_TR_WITDT* > *CTLx.DST_TR_WIDTH*
- *blk_size_bytes_dst/src_single_size_bytes* != integer

The amount of data lost is: *src_single_size_bytes* - *dst_single_size_bytes* [refer to equations (2) and (3)].

Observation: Data loss can occur when the destination is the flow controller, even if data pre-fetching is disabled, *CFGx.FCMODE* = 1.

9.2.8.2 Peripheral Interrupt Request Interface

The interface illustrated in Fig 9-37 is a simplified version of the hardware handshaking interface. In this mode:

- The interrupt line from the peripheral is tied to the *dma_req* input.
- The *dma_single* input is tied low.
- All other interface signals are ignored.

This interface can be used where the slave peripheral does not have hardware handshaking signals.

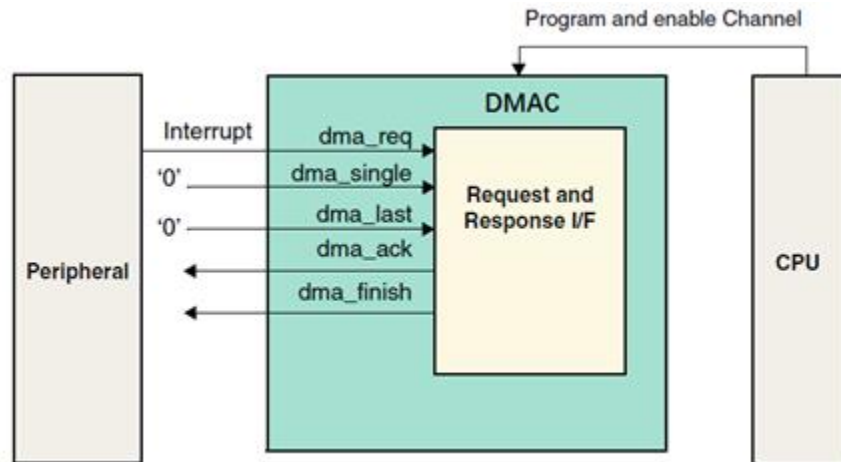


Fig 9-37 Transaction request through peripheral interrupt

The peripheral can never be the flow controller, since it cannot connect to the `dma_last` signal. The interrupt line from the peripheral is tied to the `dma_req` line, as shown in Fig 9-37. The timing of the interrupt line from the peripheral must be the same as the `dma_req` line.

Since the `dma_ack` line is not sampled by the peripheral, the handshaking loop is as follows:

- (1) Peripheral generates an interrupt that asserts `dma_req`.
- (2) DMAC completes the burst transaction and generates an end-of-burst transaction interrupt, `IntSrcTran/IntDstTran`. Interrupts must be enabled and the transaction complete interrupt unmasked.
- (3) The interrupt service routine clears the interrupt in the peripheral so that the `dma_req` is de-asserted.

Notice that `dma_single` is hardcoded to an inactive level.

9.2.9 Flow Control Configurations

Fig 9-38 indicates five different flow control configurations using hardware handshaking interfaces – a simplified version of the interface is shown.

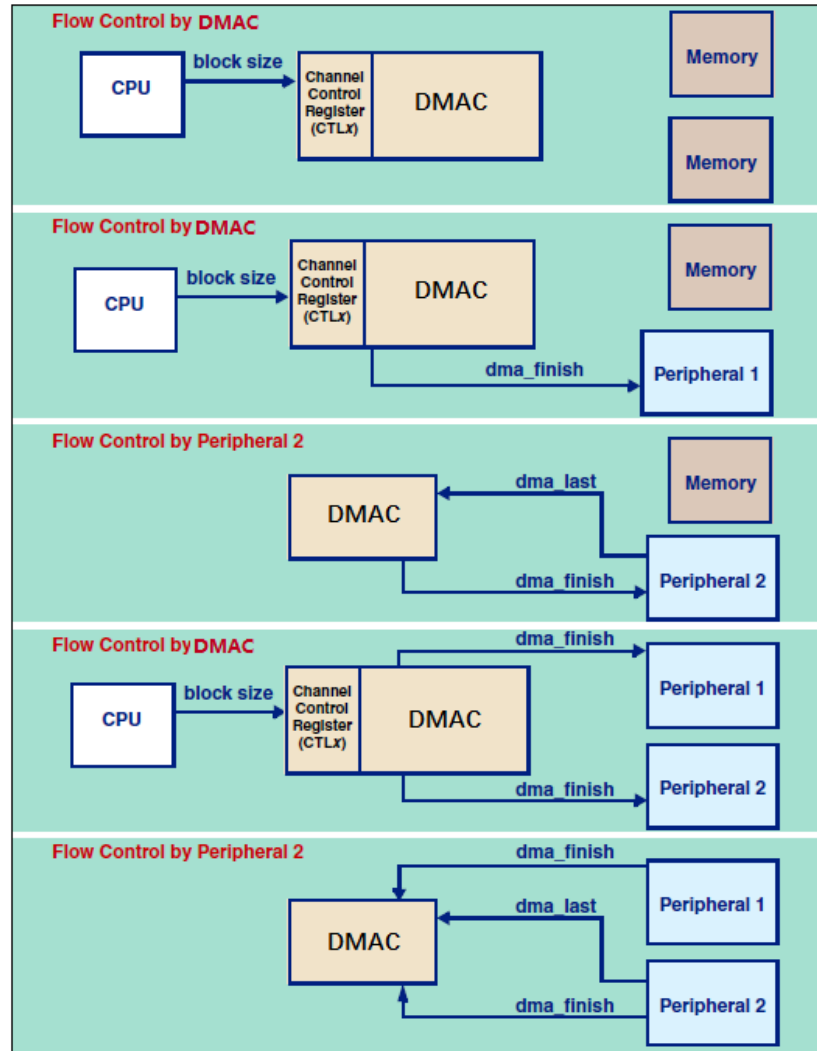


Fig 9-38 Flow control configurations

9.2.10 Peripheral Burst Transaction Requests

For a source FIFO, an active edge is triggered on dma_req when the source FIFO exceeds some watermark level. For a destination FIFO, an active edge is triggered on dma_req when the destination FIFO drops below some watermark level.

This section investigates the optimal settings of these watermark levels on the source and destination peripherals and their relationship to, respectively:

- Source transaction length, CTLx.SRC_MSIZ
- Destination transaction length, CTLx.DEST_MSIZ

For demonstration purposes, a receive SSI is used as a source peripheral, and a transmit SSI is used as a destination peripheral.

Note: Throughout this section, SSI-related parameters are prefixed with "SSI". DMAC-related parameters are prefixed with "DMA".

9.2.10.1 Transmit Watermark Level and Transmit FIFO Underflow

During SSI serial transfers, SSI transmit FIFO requests are made to the DMAC whenever the number of entries in the SSI transmit FIFO is less than or equal to the SSI Transmit Data Level Register (SSI.DMATDLR) value. This is known as the watermark level. The DMAC responds by writing a burst of data to the SSI transmit FIFO buffer, of length DMA.CTLx.DEST_MSIZ.

Data should be fetched from the DMAC often enough for the SSI transmit FIFO to continuously perform serial transfers; that is, when the SSI transmit FIFO begins to empty, another burst transaction request should be triggered. Otherwise the SSI transmit FIFO runs out of data (underflow). To prevent this condition, the user must set the watermark level correctly.

9.2.10.2 Choosing the Transmit Watermark Level

Consider an example with the following assumption:

$$\text{DMA.CTLx.DEST_MSIZE} = \text{SSI_TX_FIFO_DEPTH} - \text{SSI.DMATDLR}$$

Note: SSI_TX_FIFO_DEPTH is the SSI transmit FIFO depth. SSI.DMATDLR controls the level at which a DMAC destination burst request is made by the SSI transmit logic. It is equal to the watermark level; that is, a destination burst request is generated (active-edge of dma_req triggered) when the number of valid data entries in the SSI transmit FIFO is equal to or below this field value.

In this situation, the number of data items to be transferred in a DMAC burst is equal to the empty space in the SSI transmit FIFO. Consider two different watermark level settings.

9.2.10.2.1 Case 1: SSI.DMATDLR = 2

Fig 9-39 illustrates the watermark levels in Case 1 where SSI.DMATDLR = 2.

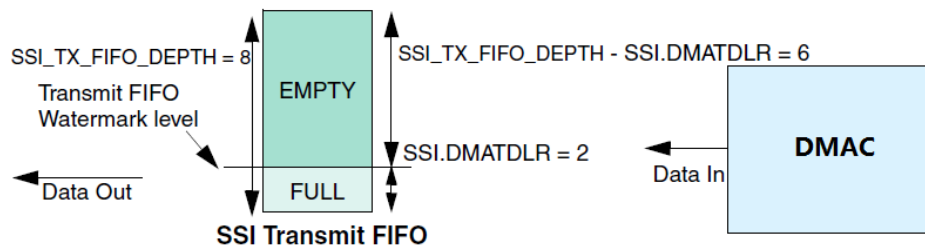


Fig 9-39 Case 1 watermark levels where SSI.DMATDLR = 2

Case 1 uses the parameters listed in Table 9-12.

Table 9-12 Transmit watermark level – Case 1

Parameter	Description
SSI.DMATDLR = 2	SSI transmit FIFO watermark level
DMA.CTLx.DEST_MSIZ = SSI_TX_FIFO_DEPTH - SSI.DMATDLR = 6	DMA.CTLx.DEST_MSIZ is equal to the empty space in the transmit FIFO at the time the burst request is made.
SSI_TX_FIFO_DEPTH = 8	SSI transmit FIFO depth
DMA.CTLx.BLOCK_TS = 30	Block size

The number of burst transactions that are needed equals the block size divided by the number of data items per burst:

$$\text{DMA.CTLx.BLOCK_TS} / \text{DMA.CTLx.DEST_MSIZ} = 30 / 6 = 5$$

The number of burst transactions in the DMAC block transfer is 5, but the watermark level, SSI.DMATDLR, is quite low. Therefore, the probability of an SSI underflow is high where the SSI serial transmit line needs to transmit data, but where there is no data left in the transmit FIFO. This occurs because the DMAC has not had time to service the DMAC request before the SSI transmit FIFO becomes empty.

9.2.10.2.2 Case 2: SSI.DMATDLR = 6

Fig 9-40 illustrates the watermark levels in Case 2 where SSI.DMATDLR = 6.

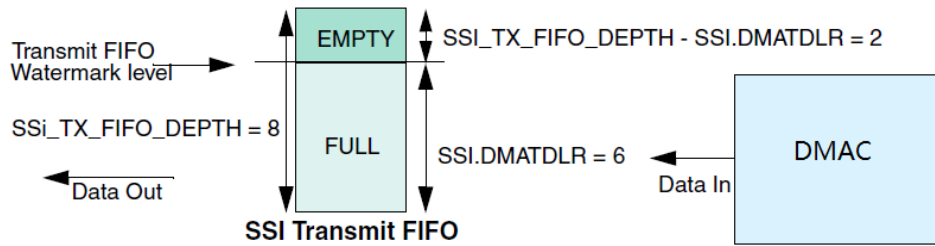


Fig 9-40 Case 2 watermark levels where SSI.DMATDLR = 6

Case 2 uses the parameters listed in Table 9-13.

Table 9-13 Transmit watermark level – Case 2

Parameter	Description
SSI.DMATDLR = 6	SSI transmit FIFO watermark level
DMA.CTLx.DEST_MSIZ = SSI_TX_FIFO_DEPTH - SSI.DMATDLR = 2	DMA.CTLx.DEST_MSIZ is equal to the empty space in the transmit FIFO at the time the burst request is made.
SSI_TX_FIFO_DEPTH = 8	SSI transmit FIFO depth
DMA.CTLx.BLOCK_TS = 30	Block size

The number of burst transactions in the block are:

$$\text{DMA.CTLx.BLOCK_TS} / \text{DMA.CTLx.DEST_MSIZE} = 30 / 2 = 15$$

In this block transfer, there are fifteen destination burst transactions in a DMA block transfer, but the watermark level, SSI.DMATDLR, is high. Therefore, the probability of an SSI underflow is low because the DMAC has plenty of time to service the destination burst transaction request before the SSI transmit FIFO becomes empty.

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This potentially provides a greater amount of bursts per block and a worse bus utilization than the former case.

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the following ratio:

$$\frac{\text{Rate of SSI data transmission}}{\text{Rate of DW_ahb_dmac response to destination burst requests}}$$

For example, promoting the channel to the highest-priority channel in the DMAC, and promoting the DMAC master interface to the highest-priority master in the AHB layer, increases the rate at which the DMAC can respond to burst transaction requests. This in turn allows the user to decrease the watermark level, which improves bus utilization without compromising the probability of an underflow occurring.

9.2.10.3 Selecting CTLx.DEST_MSIZ and Transmit FIFO Overflow

As can be seen from Fig 9-40, programming DMA.CTLx.DEST_MSIZ to a value greater than the watermark level that triggers the DMAC request may cause overflow when there is not enough space in the SSI transmit FIFO to service the destination burst request. Therefore, the following equation must be adhered to in order to avoid overflow:

$$\text{DMA.CTLx.DEST_MSIZ} \leq \text{SSI_TX_FIFO_DEPTH} - \text{SSI.DMATDLR} \quad (13)$$

In “Case 2: SSI.DMATDLR = 6”, the amount of space in the transmit FIFO at the time the burst request is made is equal to the destination burst length, DMA.CTLx.DEST_MSIZ. Thus, the transmit FIFO may be full, but not overflowed, upon completion of the burst transaction.

Therefore, for optimal operation, DMA.CTLx.DEST_MSIZ should be set at the FIFO level that triggers a transmit DMAC request; that is:

$$\text{DMA.CTLx.DEST_MSIZ} = \text{SSI_TX_FIFO_DEPTH} - \text{SSI.DMATDLR} \quad (14)$$

This is the setting used in Fig 9-28.

Adhering to equation (14) reduces the number of DMAC bursts needed for a block transfer, and this in turn improves AHB bus utilization.

Note: The SSI transmit FIFO is not full at the end of a DMAC burst transaction if the SSI has successfully transmitted one data item or more on the SSI serial transmit line before the end of the burst transaction.

9.2.10.4 Receive Watermark Level and Receive FIFO Overflow

During SSI serial transfers, SSI receive FIFO requests are made to the DMAC whenever the number of entries in the SSI receive FIFO is at or above the DMAC Receive Data Level Register; that is, SSI.DMARDLR+1. This is known as the watermark level. The DMAC responds by reading a burst of data from the receive FIFO buffer of length DMA.CTLx.SRC_MSIZEx.

Note: SSI.DMARDLR controls the level at which a source burst request is made by the receive logic. When the number of valid data entries in the SSI receive FIFO is equal to or greater than the watermark level (DMARDLR+1), a source burst request is generated (active-edge of dma_req triggered).

Data should be fetched by the DMAC often enough for the SSI receive FIFO to accept serial transfers continuously; that is, when the SSI receive FIFO begins to fill, another burst transaction request should be triggered. Otherwise, the SSI receive FIFO fills with data (overflow). To prevent this condition, the user must correctly set the watermark level.

9.2.10.5 Choosing the Receive Watermark level

Similar to choosing the transmit watermark level described earlier, the receive watermark level, SSI.DMARDLR+1, should be set to minimize the probability of overflow, as shown in Fig 9-41. It is a trade-off between the number of burst transactions required per block versus the probability of an overflow occurring.

9.2.10.6 Selecting CTLx.SRC_MSIZEx and Receive FIFO Underflow

As can be seen in Fig 9-41, programming a source burst transaction length greater than the watermark level may cause underflow when there is not enough data to service the source burst request. Therefore, equation (15) below must be adhered to in order to avoid underflow.



Fig 9-41 SSI receive FIFO

If the number of data items in the SSI receive FIFO is equal to the source burst length at the time the source burst request is made – DMA.CTLx.SRC_MSIZEx – the SSI receive FIFO may be emptied, but not underflowed, at the completion of the source burst transaction. For optimal operation, DMA.CTLx.SRC_MSIZEx should be set at the watermark level; that is:

$$\text{DMA.CTLx.SRC_MSIZE} = \text{SSI.DMARDLR} + 1 \quad (15)$$

Adhering to equation (15) reduces the number of burst transactions in a block transfer, and this in turn can improve AHB bus utilization.

Note: The SSI receive FIFO is not empty at the end of the source burst transaction if the SSI has successfully received one data item or more on the SSI serial receive line before the end of the burst, as illustrated in Fig 9-41.

9.2.11 Generating Requests for the AHB Master Bus Interface

Each channel has a source state machine and destination state machine running in parallel. These state machines generate the request inputs to the arbiter, which arbitrates for the master bus interface (one arbiter per master bus interface).

When the source/destination state machine is granted control of the master bus interface, and when the master bus interface is granted control of the external AHB bus, then AHB transfers between the peripheral and the DMAC (on behalf of the granted state machine) can take place.

AHB transfers from the source peripheral or to the destination peripheral cannot proceed until the channel FIFO is ready. For burst transaction requests and for transfers involving memory peripherals, the criterion for “FIFO readiness” is controlled by the FIFO_MODE field of the CFGx register.

The definition of FIFO readiness is the same for:

- Single transactions
- Burst transactions, where CFGx.FIFO_MODE = 0
- Transfers involving memory peripherals, where CFGx.FIFO_MODE = 0

The channel FIFO is deemed ready when the space/data available is sufficient to complete a single AHB transfer of the specified transfer width. FIFO readiness for source transfers occurs when the channel FIFO contains enough room to accept at least a single transfer of CTLx.SRC_TR_WIDTH width. FIFO readiness for destination transfers occurs when the channel FIFO contains data to form at least a single transfer of CTLx.DST_TR_WIDTH width.

Note: An exception to FIFO readiness for destination transfers occurs in “FIFO flush mode.” In this mode, FIFO readiness for destination transfers occurs when the channel FIFO contains data to form at least a single transfer of CTLx.SRC_TR_WIDTH width (and not CTLx.DST_TR_WIDTH width, as is the normal case). For an explanation of FIFO flush mode, refer to “Example 5”.

When CFGx.FIFO_MODE = 1, then the criteria for FIFO readiness for burst transaction requests and transfers involving memory peripherals are as follows:

- A FIFO is ready for a source burst transfer when the FIFO is less than half empty.
- A FIFO is ready for a destination burst transfer when the FIFO is greater than or equal to half full.

Exceptions to this “readiness” occur. During these exceptions, a value of CTLx.FIFO_MODE = 0 is assumed. The following are the exceptions:

- Near the end of a burst transaction or block transfer – The channel source state machine does not wait for the channel FIFO to be less than half empty if the number of source data items left to complete the source burst transaction or source block transfer is less than DMAH_CHx_FIFO_DEPTH/2. Similarly, the channel destination state machine does not wait for the channel FIFO to be greater than or equal to half full, if the number of destination data items left to complete the destination burst transaction or destination block transfer is less than DMAH_CHx_FIFO_DEPTH/2.
- In FIFO flush mode – For an explanation of FIFO flush mode, refer to “Example 5”.
- When a channel is suspended – The destination state machine does not wait for the FIFO to become half empty to flush the FIFO, regardless of the value of the FIFO_MODE field.
- After receipt of a split/retry response from a source or destination – The AMBA protocol requires that after an AHB master receives a split/retry response, it must re-issue the transfer that received the split/retry before attempting any other transfer. Therefore, a transfer is re-issued to the same address that returned the split/retry, regardless of FIFO_MODE, when the DMAC is next granted the AHB bus. This is repeated until an OKAY response is received on the AHB hresp bus.

When the source/destination peripheral is not memory, the source/destination state machine waits for a single/burst transaction request. Upon receipt of a transaction request and only if the channel FIFO is “ready” for source/destination AHB transfers, a request for the master bus interface is made by the source/destination state machine.

Note: There is one exception to this, which occurs when the destination peripheral is the flow controller and CFGx.FCMODE = 1 (data pre-fetching is disabled). Then the source state machine does not generate a request for the master bus interface (even if the FIFO is “ready” for source transfers and has received a source transaction request) until the destination requests new data. Refer to “Example 8”.

When the source/destination peripheral is memory, the source/destination state machine must wait until the channel FIFO is “ready.” A request is then made for the master bus interface. There is no handshaking mechanism employed between a memory peripheral and the DMAC.

9.2.11.1 Locked DMA Transfers

It is possible to program the DMAC for:

- Bus locking – Asserts the AHB hlock signal.
- Channel locking – Locks the arbitration for the AHB master interface, which grants ownership of the master bus interface to one of the requesting channel state machines (source or destination).

Bus and channel locking can proceed for the duration of a DMA transfer, a block transfer, or a single or burst transaction.

9.2.11.1.1 Bus Locking

If the LOCK_B bit in the channel configuration register (CFGx) is set, then the AHB hlock signal is asserted for the duration specified in the LOCK_B_L field.

9.2.11.1.2 Channel Locking

If the LOCK_CH field is set, then the arbitration for the master bus interface is exclusively reserved for the source and destination peripherals of that channel for the duration specified in the LOCK_CH_L field.

If bus locking is activated for a certain duration, then it follows that the channel is also automatically locked for that duration. Three cases arise:

- CFGx.LOCK_B = 0 – Programmed values of CFGx.LOCK_CH and CFGx.LOCK_CH_L are used.
- CFGx.LOCK_B = 1 and CFGx.LOCK_CH = 0 – DMA transfer proceeds as if CFGx.LOCK_CH = 1 and CFGx.LOCK_CH_L = CFGx.LOCK_B_L. The programmed values of CFGx.LOCK_CH and CFGx.LOCK_CH_L are ignored.
- CFGx.LOCK_B = 1 and CFGx.LOCK_CH = 1 – Two cases arise:
 - CFGx.LOCK_B_L ≤ CFGx.LOCK_CH_L – In this case, the DMA transfer proceeds as if CFGx.LOCK_CH_L = CFGx.LOCK_B_L and the programmed value of CFGx.LOCK_CH_L is ignored. Thus, if bus locking is enabled over the DMA transfer level, then channel locking is enabled over the DMA transfer level, regardless of the programmed value of CFGx.LOCK_CH_L.
 - CFGx.LOCK_B_L > CFGx.LOCK_CH_L – The programmed value of CFGx.LOCK_CH_L is used. Thus, if bus locking is enabled over the DMA block transfer level and channel locking is enabled over the DMA transfer level, then channel locking is performed over the DMA transfer level.

9.2.11.1.3 Locking Levels

If locking is enabled for a channel, then locking of the AHB master bus interface at a programmed locking transfer level is activated when the channel is first granted the AHB master bus interface at the start of that locking transfer level. It continues until the locking transfer level has completed; that is, if channel 0 has enabled channel level locking at the block transfer level, then this channel locks the master bus interface when it is first granted the master bus interface at the start of the block transfer, and continues to lock the master bus interface until the block transfer has completed.

Source and destination block transfers occur successively in time, and a new source block cannot commence until the previous destination block has completed. When both source and destination are on the same AHB layer, then block level locking is terminated on completion of the block to the destination. If they are on separate layers, then block-level locking is terminated on completion of the block on that layer—when the source block on the source AHB layer completes, and when the destination block on the destination AHB layer completes. The same is true for DMA transfer-level locking.

Transaction-level locking is different due to the fact that source and destination transactions occur independently in time, and the number of source and destination transactions in a DMA block or DMA transfer do not have to match. When the source and destination are on the same AHB layer, then transaction-level locking is cleared at the end of a source or destination transaction only if the opposing peripheral is not currently in the middle of a transaction.

For example, if locking is enabled at the transaction level and an end-of-source transaction is signaled, then this disables locking only if one of the following is true:

- The destination is on a different AHB layer
- The destination is on the same AHB layer, but the channel is not currently in the middle of a transaction to the destination peripheral.

The same rules apply when an end-of-destination transaction is signalled.

If channel-level or bus-level locking is enabled for a channel at the transaction level, and either the source or destination of the channel is a memory device, then the locking is ignored and the channel proceeds as if locking (bus or channel) is disabled.

Note: Since there is no notion of a transaction level for a memory peripheral, then transaction-level locking is not allowed when either source or destination is memory.

9.2.11.1.4 Channel Locking and Deadlock

Certain combinations of channel-level and bus-level locking may lead to deadlock, where multiple channels are concurrently enabled and no channel can proceed with the DMA transfer. This occurs only for configurations where `DMAH_NUM_MASTER_INT > 1` and `DMAH_NUM_CHANNELS > 1`.

9.2.12 Arbitration for AHB Master Interface

Each DMAC channel has two request lines that request ownership of a particular master bus interface: channel source and channel destination request lines.

Source and destination arbitrate separately for the bus. Once a source/destination state machine gains ownership of the master bus interface and the master bus interface has ownership of the AHB bus, then AHB transfers can proceed between the peripheral and DMAC. Fig 9-42 illustrates the arbitration flow of the master bus interface.

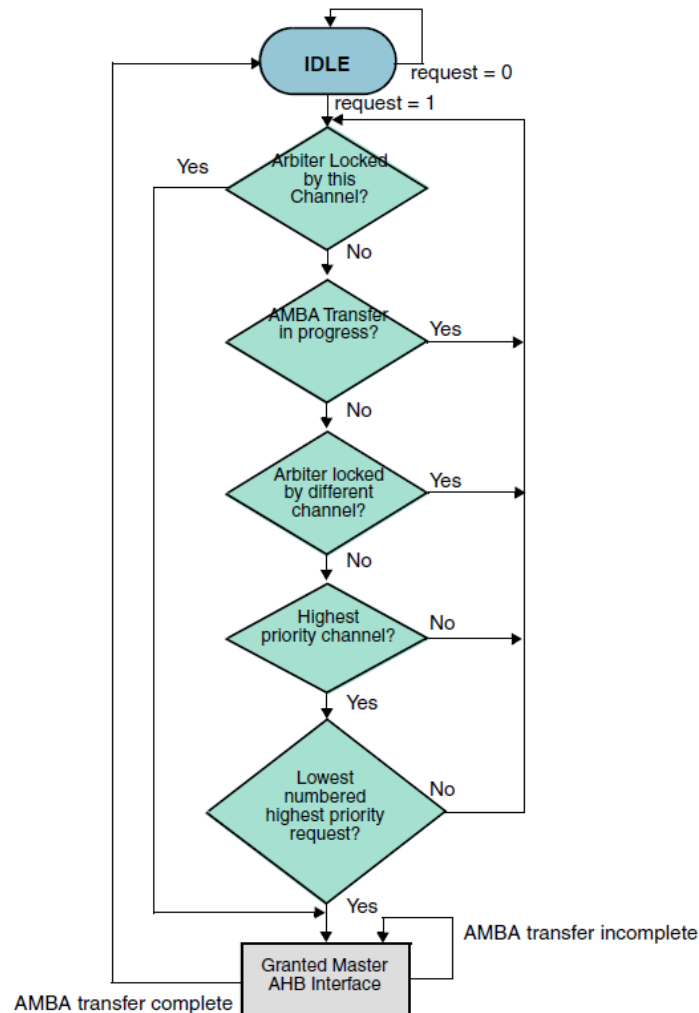


Fig 9-42 Arbitration flow for master bus interface

An arbitration scheme decides which of the request lines ($2 * \text{DMAH_NUM_CHANNELS}$) is granted the particular master bus interface. Each channel has a programmable priority. A request for the master bus interface can be made at any time, but is granted only after the current AHB transfer (burst or single) has completed. Therefore, if the master interface is transferring data for a lower priority channel and a higher priority channel requests service, then the master interface will complete the current burst for the lower priority channel before switching to transfer data for the higher priority channel.

To prevent a channel from saturating the master bus interface, it can be given a maximum AMBA burst length (`MAX_ABRST` field in `CFGx` register) at channel setup time. This also prevents the master bus interface from saturating the AHB bus where the system arbiter cannot change the grant lines until the end of an undefined length burst.

The following is the interface arbitration scheme employed when no channel has locked (Channel Locking) the arbitration for the master bus interface:

- If only one request line is active at the highest priority level, then the request with the highest priority wins ownership of the AHB master bus interface; it is not necessary for the priority levels to be unique.
If more than one request is active at the highest requesting priority, then these competing requests proceed to a second tier of arbitration.
- If equal priority requests occur, then the lower-numbered channel is granted.
In other words, if a peripheral request attached to Channel 7 and a peripheral request attached to Channel 8 have the same priority, then the peripheral attached to Channel 7 is granted first.

Note: A channel source is granted before the destination if both have their request lines asserted when a grant decision is made. A channel source and channel destination inherit their channel priority and therefore always have the same priority.

9.2.13 Scatter/Gather

Scatter is relevant to a destination transfer. The destination address is incremented or decremented by a programmed amount – the scatter increment – when a scatter boundary is reached. Fig 9-43 shows an example destination scatter transfer. The destination address is incremented or decremented by the value stored in the destination scatter increment (`DSRx.DSI`) field, multiplied by the number of bytes in a single AHB transfer to the destination $\frac{\text{decoded value of } \text{CTLx.DST_TR_WIDTH}}{8}$ – when a scatter boundary is reached. The number of destination transfers between successive scatter boundaries is programmed into the Destination Scatter Count (`DSC`) field of the `DSRx` register.

Scatter is enabled by writing a 1 to the `CTLx.DST_SCATTER_EN` field. The `CTLx.DINC` field determines if the address is incremented, decremented, or remains fixed when a scatter boundary is reached. If the `CTLx.DINC` field indicates a fixed-address control throughout a DMA transfer, then the `CTLx.DST_SCATTER_EN` field is ignored, and the scatter feature is automatically disabled.

Gather is relevant to a source transfer. The source address is incremented or decremented by a programmed amount when a gather boundary is reached. The number of source transfers between successive gather boundaries is programmed into the Source Gather Count (`SGRx.SGC`) field. The source address is incremented or decremented by the value stored in the source gather increment (`SGRx.SGI`) field, multiplied by the number of bytes in a single AHB transfer from the source – $\frac{\text{decoded value of } \text{CTLx.SRC_TR_WIDTH}}{8}$ – when a gather boundary is reached.

Gather is enabled by writing a 1 to the `CTLx.SRC_GATHER_EN` field. The `CTLx.SINC` field determines if the address is incremented, decremented, or remains fixed when a gather boundary is reached. If the `CTLx.SINC` field indicates a fixed-address control throughout a DMA transfer, then the `CTLx.SRC_GATHER_EN` field is ignored, and the gather feature is automatically disabled.

Note: For multi-block transfers, the counters that keep track of the number of transfers left to reach a gather/scatter boundary are re-initialized to the source gather count (`SGRx.SGC`) and destination scatter count (`DSC`), respectively, at the start of each block transfer.

Fig 9-43 shows an example of a destination scatter transfer.

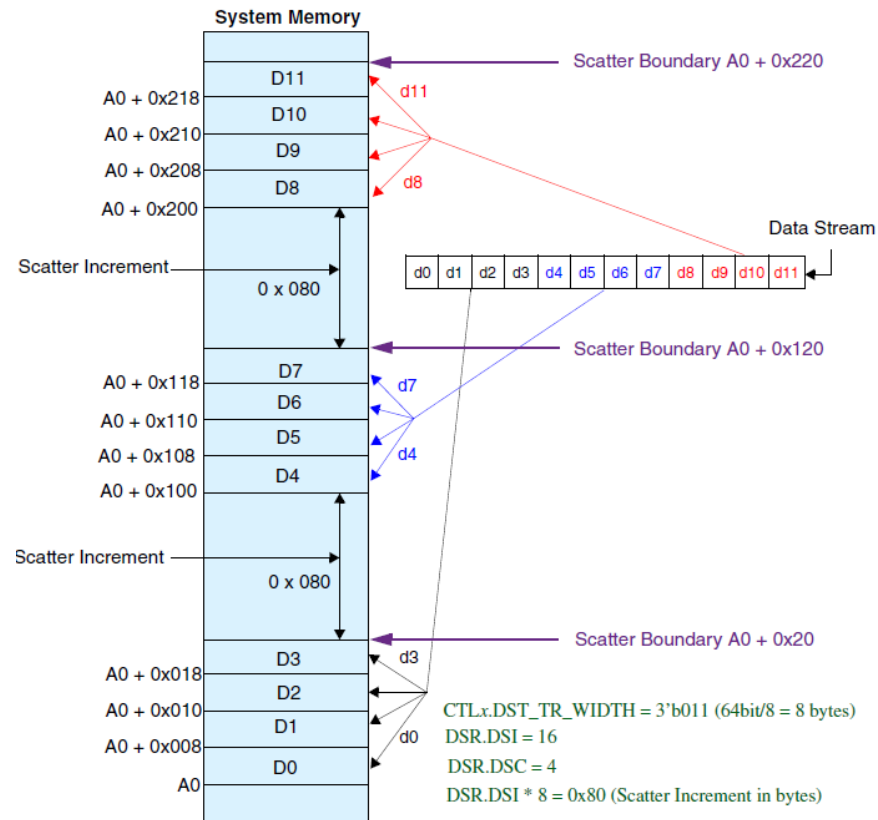


Fig 9-43 Example of destination scatter transfer

As an example of gather increment, consider the following:

SRC_TR_WIDTH = 3'b 010 (32 bit)

SGR.SGC = 0x04 (source gather count)

CTLX.SRC_GATHER_EN = 1 (source gather enabled)

SARx = A0 (starting source address)

Fig 9-44 shows a source gather when SGR.SGI = 0x01.

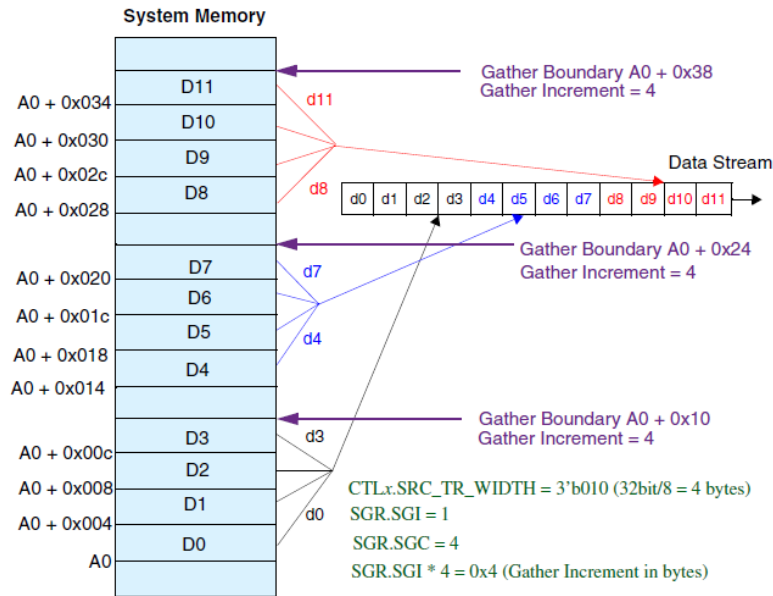


Fig 9-44 Source gather when SGR.SGI = 0x1

In general, if the starting address is A0 and CTLx.SINC = 2'b00 (increment source address control), then the transfer will be:

$A0, A0 + TWB, A0 + 2 * TWB, \dots, (A0 + (SGR.SGC - 1) * TWB) \leftarrow \text{scatter_increment} \rightarrow (A0 + (SGR.SGC * TWB) + (SGR.SGI * TWB))$

where TWB is the transfer width in bytes, decoded value of $CTLx.SRC_TR_WIDTH / 8 = \text{src_single_size_bytes}$.

9.2.14 Endianness

You can configure the endianness of the slave and master interfaces either statically through coreConsultant or dynamically through the I/O interface.

9.2.14.1 Static Endian Configuration

You can statically configure endianness under the following conditions:

- Endianness of the AHB slave interface and all configured AHB master interfaces are the same
- Endianness is known at system configuration time

Under these circumstances, you do the following:

- (1) Set the DMAH_STATIC_ENDIAN_SELECT parameter to True.
- (2) Configure the DMAC_BIG_ENDIAN parameter to either big-endian or little-endian.

9.2.14.2 Dynamic Endian Configuration

You should dynamically configure endianness if either of the following conditions exists:

- Endianness of the AHB slave interface and all configured master interfaces are not all the same
- Endianness of the slave interface or any master interface is not known at system configuration time

Under either of these circumstances, do the following:

- (1) Set the DMAH_STATIC_ENDIAN_SELECT parameter to False.
- (2) Drive the appropriate I/O pin to either big-endian or little-endian.

For example, if you were to drive dmah_big_endian_mN to 0—where N is the master number—the AMBA layer to which that master interface is attached will be little-endian. Conversely, if you were to drive dmah_big_endian_mN to 1, that master interface would be big-endian.

The dma_big_endian_slv signal works in the same way for the slave interface.

9.2.15 AHB Transfer Error Handling

Upon occurrence of an error in an AHB transfer, the following occurs:

- (1) DMA transfer in progress stops immediately
- (2) Relevant channel is disabled,
- (3) An interrupt is issued (if not masked)

If multiple channels are enabled, only the one where the AHB error was detected is disabled. The contents of the FIFO are not cleared, but they become inaccessible and are overwritten once the channel is re-enabled to start a new sequence.

There is no support for automatically resuming the transfer from the point where the error occurred, and the full block transfer has to be re-initiated in order to be successfully completed.

The DMA does not use the hardware handshaking interface to signal the error occurrence in any way, nor does it signal the end of a transfer. In practice, this means that if a request from a peripheral is active when the error occurs—`dma_req` is high if peripheral is the flow controller; `dma_req` or `dma_single` is high if peripheral is not the flow controller—the channel is disabled without the DMA ever asserting `dma_ack` (or `dma_finish`).

The hardware handshake interface on the peripheral side has to be re-initiated by the CPU upon detection of the error interrupt. The `dma_req` signal needs to be brought low before the channel is re-enabled and then brought high when the channel has been enabled.

9.3 Registers

This section describes the programmable registers of the DMAC.

Note: There are references to both software and hardware parameters throughout this chapter. The software parameters are the field names in each register description table and are prefixed by the register name; for example, the Block Transfer Size field in the Control register for Channel x is designated as “`CTLx.BLOCK_TS`”.

Shipped with the DMAC component is an address definition (memory map) C header file. This can be used when the DMAC is programmed in a C environment.

9.3.1 Register Memory Map

Table 3-2 shows the memory map for the DMAC.

Note: The address offsets of the registers are always fixed, regardless of the number of configured channels.

Table 9-14 Memory map of DMAC

Name	Address Offset	Access	Reset	Description
Channel Registers				
If <code>DMAH_NUM_CHANNELS = "dnc"</code> , “Reg Exist” field shows “dnc” value for register to exist.				
SAR0	0x000	R/W	0x0	Channel 0 Source Address Register Reg Exist: Yes
DAR0	0x008	R/W	0x0	Channel 0 Destination Address Register Reg Exist: Yes
LLP0	0x010	R/W	0x0	Channel 0 Linked List Pointer Register Reg Exist: <code>llp0_hc = False</code> <code>llp0_hc = DMAH_CH0_HC_LLP</code>
CTL0	0x018	R/W	Configuration dependent	Channel 0 Control Register Reg Exist: Yes
SSTAT0	0x020	R/W	0x0	Channel 0 Source Status Register Reg Exist: <code>sstat0 = True</code> <code>sstat0 = DMAH_CH0_STAT_SRC</code>
DSTAT0	0x028	R/W	0x0	Channel 0 Destination Status Register

				Reg Exist: dstat0 = True dstat0 = DMAH_CH0_STAT_DST
SSTATAR0	0x030	R/W	0x0	Channel 0 Source Status Address Register Reg Exist: stat0 = True stat0 = DMAH_CH0_STAT_SRC
DSTATAR0	0x038	R/W	0x0	Channel 0 Destination Status Address Register Reg Exist: dstat0 = True dstat0 = DMAH_CH0_STAT_DST
CFG0	0x040	R/W	0x00000000400000e00	Channel 0 Configuration Register Reg Exist: Yes
SGR0	0x048	R/W	0x0	Channel 0 Source Gather Register Reg Exist: sgr0 = True sgr0 = DMAH_CH0_SRC_GAT_EN
DSR0	0x050	R/W	0x0	Channel 0 Destination Scatter Register Reg Exist: dsr0 = True dsr0 = DMAH_CH0_DST_SCA_EN
SAR1	0x058	R/W	0x0	Channel 1 Source Address Register Reg Exist: dnc ≥ 2
DAR1	0x060	R/W	0x0	Channel 1 Destination Address Register Reg Exist: dnc ≥ 2
LLP1	0x068	R/W	0x0	Channel 1 Linked List Pointer Register Reg Exist: dnc ≥ 2 and llp1_hc = False
CTL1	0x070	R/W	Configuration dependent	Channel 1 Control Register Reg Exist: dnc ≥ 2
SSTAT1	0x078	R/W	0x0	Channel 1 Source Status Register Reg Exist: dnc ≥ 2 and sstat1 = True
DSTAT1	0x080	R/W	0x0	Channel 1 Destination Status Register Reg Exist: dnc ≥ 2 and dstat1 = True
SSTATAR1	0x088	R/W	0x0	Channel 1 Source Status Address Register Reg Exist: dnc ≥ 2 and sstat1 = True
DSTATAR1	0x090	R/W	0x0	Channel 1 Destination Status Address Register Reg Exist: dnc ≥ 2 and dstat1 = True
CFG1	0x098	R/W	0x00000000400000e20	Channel 1 Configuration Register Reg Exist: dnc ≥ 2
SGR1	0x0a0	R/W	0x0	Channel 1 Source Gather Register Reg Exist: dnc ≥ 2 and sgr1 = True
DSR1	0x0a8	R/W	0x0	Channel 1 Destination Scatter Register Reg Exist: dnc ≥ 2 and dsr1 = True
SAR2	0x0b0	R/W	0x0	Channel 2 Source Address Register Reg Exist: dnc ≥ 3
DAR2	0x0b8	R/W	0x0	Channel 2 Destination Address Register Reg Exist: dnc ≥ 3
LLP2	0x0c0	R/W	0x0	Channel 2 Linked List Pointer Register Reg Exist: dnc ≥ 3 and llp2_hc = False
CTL2	0x0c8	R/W	Configuration dependent	Channel 2 Control Register Reg Exist: dnc ≥ 3
SSTAT2	0x0d0	R/W	0x0	Channel 2 Source Status Register Reg Exist: dnc ≥ 3 and sstat2 = True
DSTAT2	0x0d8	R/W	0x0	Channel 2 Destination Status Register Reg Exist: dnc ≥ 3 and dstat2 = True
SSTATAR2	0x0e0	R/W	0x0	Channel 2 Source Status Address Register Reg Exist: dnc ≥ 3 and sstat2 = True
DSTATAR2	0x0e8	R/W	0x0	Channel 2 Destination Status Address Register Reg Exist: dnc ≥ 3 and dstat2 = True
CFG2	0x0f0	R/W	0x00000000400000e40	Channel 2 Configuration Register Reg Exist: dnc ≥ 3

SGR2	0x0f8	R/W	0x0	Channel 2 Source Gather Register Reg Exist: dnc ≥ 3 and sgr2 = True
DSR2	0x100	R/W	0x0	Channel 2 Destination Scatter Register Reg Exist: dnc ≥ 3 and dsr2 = True
SAR3	0x108	R/W	0x0	Channel 3 Source Address Register Reg Exist: dnc ≥ 4
DAR3	0x110	R/W	0x0	Channel 3 Destination Address Register Reg Exist: dnc ≥ 4
LLP3	0x118	R/W	0x0	Channel 3 Linked List Pointer Register Reg Exist: dnc ≥ 4 and llp3_hc = False
CTL3	0x120	R/W	Configuration dependent	Channel 3 Control Register Reg Exist: dnc ≥ 4
SSTAT3	0x128	R/W	0x0	Channel 3 Source Status Register Reg Exist: dnc ≥ 4 and sstat3 = True
DSTAT3	0x130	R/W	0x0	Channel 3 Destination Status Register Reg Exist: dnc ≥ 4 and dstat3 = True
SSTATAR3	0x138	R/W	0x0	Channel 3 Source Status Address Register Reg Exist: dnc ≥ 4 and sstat3 = True
DSTATAR3	0x140	R/W	0x0	Channel 3 Destination Status Address Register Reg Exist: dnc ≥ 4 and dstat3 = True
CFG3	0x148	R/W	0x0000000400000e40	Channel 3 Configuration Register Reg Exist: dnc ≥ 4
SGR3	0x150	R/W	0x0	Channel 3 Source Gather Register Reg Exist: dnc ≥ 4 and sgr3 = True
DSR3	0x158	R/W	0x0	Channel 3 Destination Scatter Register Reg Exist: dnc ≥ 4 and dsr3 = True
SAR4	0x160	R/W	0x0	Channel 4 Source Address Register Reg Exist: dnc ≥ 5
DAR4	0x168	R/W	0x0	Channel 4 Destination Address Register Reg Exist: dnc ≥ 5
LLP4	0x170	R/W	0x0	Channel 4 Linked List Pointer Register Reg Exist: dnc ≥ 5 and llp4_hc = False
CTL4	0x178	R/W	Configuration dependent	Channel 4 Control Register Reg Exist: dnc ≥ 5
SSTAT4	0x180	R/W	0x0	Channel 4 Source Status Register Reg Exist: dnc ≥ 5 and sstat4 = True
DSTAT4	0x188	R/W	0x0	Channel 4 Destination Status Register Reg Exist: dnc ≥ 5 and dstat4 = True
SSTATAR4	0x190	R/W	0x0	Channel 4 Source Status Address Register Reg Exist: dnc ≥ 5 and sstat4 = True
DSTATAR4	0x198	R/W	0x0	Channel 4 Destination Status Address Register Reg Exist: dnc ≥ 5 and dstat4 = True
CFG4	0x1a0	R/W	0x0000000400000e80	Channel 4 Configuration Register Reg Exist: dnc ≥ 5
SGR4	0x1a8	R/W	0x0	Channel 4 Source Gather Register Reg Exist: dnc ≥ 5 and sgr4 = True
DSR4	0x1b0	R/W	0x0	Channel 4 Destination Scatter Register Reg Exist: dnc ≥ 5 and dsr4 = True
SAR5	0x1b8	R/W	0x0	Channel 5 Source Address Register Reg Exist: dnc ≥ 6
DAR5	0x1c0	R/W	0x0	Channel 5 Destination Address Register Reg Exist: dnc ≥ 6
LLP5	0x1c8	R/W	0x0	Channel 5 Linked List Pointer Register Reg Exist: dnc ≥ 6 and llp5_hc = False
CTL5	0x1d0	R/W	Configuration dependent	Channel 5 Control Register Reg Exist: dnc ≥ 6

SSTAT5	0x1d8	R/W	0x0	Channel 5 Source Status Register Reg Exist: dnc ≥ 6 and sstat5 = True
DSTAT5	0x1e0	R/W	0x0	Channel 5 Destination Status Register Reg Exist: dnc ≥ 6 and dstat5 = True
SSTATAR5	0x1e8	R/W	0x0	Channel 5 Source Status Address Register Reg Exist: dnc ≥ 6 and sstat5 = True
DSTATAR5	0x1f0	R/W	0x0	Channel 5 Destination Status Address Register Reg Exist: dnc ≥ 6 and dstat5 = True
CFG5	0x1f8	R/W	0x0000000400000ea0	Channel 5 Configuration Register Reg Exist: dnc ≥ 6
SGR5	0x200	R/W	0x0	Channel 5 Source Gather Register Reg Exist: dnc ≥ 6 and sgr5 = True
DSR5	0x208	R/W	0x0	Channel 5 Destination Scatter Register Reg Exist: dnc ≥ 6 and dsr5 = True
SAR6	0x210	R/W	0x0	Channel 6 Source Address Register Reg Exist: dnc ≥ 7
DAR6	0x218	R/W	0x0	Channel 6 Destination Address Register Reg Exist: dnc ≥ 7
LLP6	0x220	R/W	0x0	Channel 6 Linked List Pointer Register Reg Exist: dnc ≥ 7 and llp6_hc = False
CTL6	0x228	R/W	Configuration dependent	Channel 6 Control Register Reg Exist: dnc ≥ 7
SSTAT6	0x230	R/W	0x0	Channel 6 Source Status Register Reg Exist: dnc ≥ 7 and sstat6 = True
DSTAT6	0x238	R/W	0x0	Channel 6 Destination Status Register Reg Exist: dnc ≥ 7 and dstat6 = True
SSTATAR6	0x240	R/W	0x0	Channel 6 Source Status Address Register Reg Exist: dnc ≥ 7 and sstat6 = True
DSTATAR6	0x248	R/W	0x0	Channel 6 Destination Status Address Register Reg Exist: dnc ≥ 7 and dstat6 = True
CFG6	0x250	R/W	0x0000000400000ec0	Channel 6 Configuration Register Reg Exist: dnc ≥ 7
SGR6	0x258	R/W	0x0	Channel 6 Source Gather Register Reg Exist: dnc ≥ 7 and sgr6 = True
DSR6	0x260	R/W	0x0	Channel 6 Destination Scatter Register Reg Exist: dnc ≥ 7 and dsr6 = True
SAR7	0x268	R/W	0x0	Channel 7 Source Address Register Reg Exist: dnc = 8
DAR7	0x270	R/W	0x0	Channel 7 Destination Address Register Reg Exist: dnc = 8
LLP7	0x278	R/W	0x0	Channel 7 Linked List Pointer Register Reg Exist: dnc = 8 and llp7_hc = False
CTL7	0x280	R/W	Configuration dependent	Channel 7 Control Register Reg Exist: dnc = 8
SSTAT7	0x288	R/W	0x0	Channel 7 Source Status Register Reg Exist: dnc = 8 and sstat7 = True
DSTAT7	0x290	R/W	0x0	Channel 7 Destination Status Register Reg Exist: dnc = 8 and dstat7 = True
SSTATAR7	0x298	R/W	0x0	Channel 7 Source Status Address Register Reg Exist: dnc = 8 and sstat7 = True
DSTATAR7	0x2a0	R/W	0x0	Channel 7 Destination Status Address Register Reg Exist: dnc = 8 and dstat7 = True
CFG7	0x2a8	R/W	0x0000000400000ee0	Channel 7 Configuration Register Reg Exist: dnc = 8
SGR7	0x2b0	R/W	0x0	Channel 7 Source Gather Register Reg Exist: dnc = 8 and sgr7 = True

DSR7	0x2b8	R/W	0x0	Channel 7 Destination Scatter Register Reg Exist: dnc = 8 and dsr7= True
Interrupt Registers RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr				
RawTfr	0x2c0	R	0x0	Raw Status for IntTfr Interrupt Reg Exist: Yes
RawBlock	0x2c8	R	0x0	Raw Status for IntBlock Interrupt Reg Exist: Yes
RawSrcTran	0x2d0	R	0x0	Raw Status for IntSrcTran Interrupt Reg Exist: Yes
RawDstTran	0x2d8	R	0x0	Raw Status for IntDstTran Interrupt Reg Exist: Yes
RawErr	0x2e0	R	0x0	Raw Status for IntErr Interrupt Reg Exist: Yes
StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr				
StatusTfr	0x2e8	R	0x0	Status for IntTfr Interrupt Reg Exist: Yes
StatusBlock	0x2f0	R	0x0	Status for IntBlock Interrupt Reg Exist: Yes
StatusSrcTran	0x2f8	R	0x0	Status for IntSrcTran Interrupt Reg Exist: Yes
StatusDstTran	0x300	R	0x0	Status for IntDstTran Interrupt Reg Exist: Yes
StatusErr	0x308	R	0x0	Status for IntErr Interrupt Reg Exist: Yes
MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr				
MaskTfr	0x310	R/W	0x0	Mask for IntTfr Interrupt Reg Exist: Yes
MaskBlock	0x318	R/W	0x0	Mask for IntBlock Interrupt Reg Exist: Yes
MaskSrcTran	0x320	R/W	0x0	Mask for IntSrcTran Interrupt Reg Exist: Yes
MaskDstTran	0x328	R/W	0x0	Mask for IntDstTran Interrupt Reg Exist: Yes
MaskErr	0x330	R/W	0x0	Mask for IntErr Interrupt Reg Exist: Yes
ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr				
ClearTfr	0x338	W	0x0	Clear for IntTfr Interrupt Reg Exist: Yes
ClearBlock	0x340	W	0x0	Clear for IntBlock Interrupt Reg Exist: Yes
ClearSrcTran	0x348	W	0x0	Clear for IntSrcTran Interrupt Reg Exist: Yes
ClearDstTran	0x350	W	0x0	Clear for IntDstTran Interrupt Reg Exist: Yes
ClearErr	0x358	W	0x0	Clear for IntErr Interrupt Reg Exist: Yes
StatusInt	0x360	W	0x0	Status for each interrupt type Reg Exist: Yes
Miscellaneous Registers				
DmaCfgReg	0x398	R/W	0x0	DMA Configuration Register Reg Exist: Yes
ChEnReg	0x3a0	R/W	0x0	DMA Channel Enable Register Reg Exist: Yes
DmaIdReg	0x3a8	R	0x0	DMA ID Register Reg Exist: Yes

DmaTestReg	0x3b0	R/W	0x0	DMA Test Register Reg Exist: Yes
Reserved	0x3b8	N/A	-	Reserved
Reserved	0x3c0	N/A	-	Reserved
DMA_COMP_P ARAMS_6	0x3c8	R	Depends on user configuration	Refer to the bit table in the description for DMA_COMP_PARAMS_6. Reg Exist: Dependent on setting of DMAH_ADD_ENCODED_PARAMS configuration parameter. The register exists when this parameter is set to True.
DMA_COMP_P ARAMS_5	0x3d0	R	Depends on user configuration	Refer to the bit table in the description for DMA_COMP_PARAMS_5. Reg Exist: Dependent on setting of DMAH_ADD_ENCODED_PARAMS configuration parameter. The register exists when this parameter is set to True.
DMA_COMP_P ARAMS_4	0x3d8	R	Depends on user configuration	Refer to the bit table in the description for DMA_COMP_PARAMS_4. Reg Exist: Dependent on setting of DMAH_ADD_ENCODED_PARAMS configuration parameter. The register exists when this parameter is set to True.
DMA_COMP_P ARAMS_3	0x3e0	R	Depends on user configuration	Refer to the bit table in the description for DMA_COMP_PARAMS_3. Reg Exist: Dependent on setting of DMAH_ADD_ENCODED_PARAMS configuration parameter. The register exists when this parameter is set to True.
DMA_COMP_P ARAMS_2	0x3e8	R	Depends on user configuration	Refer to the bit table in the description for DMA_COMP_PARAMS_2. Reg Exist: Dependent on setting of DMAH_ADD_ENCODED_PARAMS configuration parameter. The register exists when this parameter is set to True.
DMA_COMP_P ARAMS_1	0x3f0	R	Depends on user configuration	Refer to the bit table in the description for DMA_COMP_PARAMS_1 Reg Exist: Dependent on setting of DMAH_ADD_ENCODED_PARAMS configuration parameter. The register exists when this parameter is set to True.
DMA Component ID Register	0x3f8	R	See description	Component version register. Please refer to the bit table in the description for DMA Component ID Register. Reg Exist: Yes

9.3.2 Registers and Field Descriptions

The following sections contain the memory diagrams and field descriptions for the individual registers.

9.3.2.1 Configuration and Channel Enable Registers

The channel registers consist of the following, where x = 0 to 7:

- DmaCfgReg – Configuration Register
- ChEnReg – Channel Enable Register

9.3.2.1.1 DmaCfgReg

- **Name:** DMA Configuration Register
- **Size:** 64 bits
- **Address offset:** 0x398
- **Read/write access:** read/write

This register is used to enable the DMAC, which must be done before any channel activity can begin.

Bit	Name	Access	Reset	Description
63:1	RSVD	N/A	0x0	Reserved
0	DMA_EN	R/W	0x0	DMAC Enable bit. 0 = DMAC Disabled 1 = DMAC Enabled

If the global channel enable bit is cleared while any channel is still active, then DmaCfgReg.DMA_EN still returns 1 to indicate that there are channels still active until hardware has terminated all activity on all channels, at which point the DmaCfgReg.DMA_EN bit returns 0.

9.3.2.1.2 ChEnReg

- **Name:** DMA Channel Enable Register
- **Size:** 64 bits
- **Address offset:** 0x3a0
- **Read/write access:** read/write

This is the DMAC Channel Enable Register. If software needs to set up a new channel, then it can read this register in order to find out which channels are currently inactive; it can then enable an inactive channel with the required priority.

Bit	Name	Access	Reset	Description
63:8+dnc ¹	RSVD	N/A	0x0	Reserved
7+dnc ¹ :8	CH_EN_WE	W	0x0	Channel Enable Write Enable.
7:dnc ^{1,2}	RSVD	N/A	0x0	Reserved
dnc ¹ -1:0	CH_EN	R/W	0x0	Enables/Disables the channel. Setting this bit enables a channel; clearing this bit disables the channel. 0 = Disable the Channel 1 = Enable the Channel The ChEnReg.CH_EN bit is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer.

9.3.2.2 Channel Registers

The channel registers consist of the following, where x = 0 to 7:

- CFGx – Configuration register for channel x
- CTLx – Control register for channel x
- DARx – Destination address register for channel x
- DSRx – Destination scatter register for channel x
- DSTATx – Destination status register for channel x
- DSTATARx – Destination status address register for channel x
- LLPx – Linked list pointer register for channel x
- SARx – Source address register for channel x
- SGRx – Source gather register for channel x
- SSTATx – Source status register for channel x
- SSTATARx – Source status address register for channel x

The SARx, DARx, LLPx, CTLx, and CFGx channel registers should be programmed prior to enabling the channel. However, if an LLI update occurs before commencing data transfer, SARx and DARx may not need to be programmed prior to enabling the channel; refer to rows 6 to 10 in Table 9-19. It is an Illegal Register Access when a write to the SARx, DARx, LLPx, CTLx, SSTATx, DSTATx, SSTATARx, DSTATARx, SGRx, or DSRx registers occurs when the channel is enabled.

¹ dnc = DMAH_NUM_CHANNELS

² If dnc = 8, then this field is not present.

9.3.2.2.1 SARx

- **Name:** Source Address Register for Channel x
- **Size:** 64 bits (upper 32 bits are reserved)
- **Address offset:** for x = 0 to 7:
 - SAR0 – 0x000
 - SAR1 – 0x058
 - SAR2 – 0x0b0
 - SAR3 – 0x108
 - SAR4 – 0x160
 - SAR5 – 0x1b8
 - SAR6 – 0x210
 - SAR7 – 0x268
- **Read/write access:** read/write

The starting source address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the source address of the current AHB transfer.

Note: You must program the SAR address to be aligned to CTLx.SRC_TR_WIDTH.

For information on how the SARx is updated at the start of each DMA block for multi-block transfers, refer to Table 9-19.

Bit	Name	Access	Reset	Description
63:32	RSVD	N/A	0x0	Reserved
31:0	SAR	R/W	0x0	Current Source Address of DMA transfer. Updated after each source transfer. The SINC field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every source transfer throughout the block transfer.

9.3.2.2.2 DARx

- **Name:** Destination Address Register for Channel x
- **Size:** 64 bits (upper 32 bits are reserved)
- **Address offset:** for x = 0 to 7:
 - DAR0 – 0x008
 - DAR1 – 0x060
 - DAR2 – 0x0b8
 - DAR3 – 0x110
 - DAR4 – 0x168
 - DAR5 – 0x1c0
 - DAR6 – 0x218
 - DAR7 – 0x270
- **Read/write access:** read/write

The starting destination address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the destination address of the current AHB transfer.

Note: You must program the DAR address to be aligned to CTLx.DST_TR_WIDTH.

For information on how the DARx is updated at the start of each DMA block for multi-block transfers, refer to Table 9-19.

Bit	Name	Access	Reset	Description
63:32	RSVD	N/A	0x0	Reserved
31:0	DAR	R/W	0x0	Current Destination Address of DMA transfer. Updated after each destination transfer. The DINC field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every destination transfer throughout the block transfer.

9.3.2.2.3 Hardware Realignment of SAR/DAR Registers

In a particular circumstance, during contiguous multi-block DMA transfers, the destination address can become misaligned between the end of one block and the start of the next block. When this situation occurs, DMAC re-aligns the destination address before the start of the next block.

Consider the following example. If the block length is 9, the source transfer width is 16 (halfword), and the destination transfer width is 32 (word)—the destination is programmed for contiguous block transfers— then the destination performs four word transfers followed by a halfword transfer to complete the block transfer to the destination. At the end of the destination block transfer, the address is aligned to a 16-bit transfer as the last AMBA transfer is halfword. This is misaligned to the programmed transfer size of 32 bits for the destination. However, for contiguous destination multi-block transfers, DMAC re-aligns the DAR address to the nearest 32-bit address (next 32-bit address upwards if address control is incrementing or next address downwards if address control is decrementing).

The destination address is automatically realigned by the DMAC in the following DMA transfer setup scenario:

- Contiguous multi-block transfers on destination side, AND
- $DST_TR_WIDTH > SRC_TR_WIDTH$, AND
- $(BLOCK_TS * SRC_TR_WIDTH) / DST_TR_WIDTH \neq \text{integer}$ (where SRC_TR_WIDTH , DST_TR_WIDTH is byte width of transfer)

9.3.2.2.4 LLPx

- **Name:** Linked List Pointer Register for Channel x
- **Size:** 64 bits (upper 32 bits are reserved)
- **Address offset:** for x = 0 to 7:
 - LLP0 – 0x010
 - LLP1 – 0x068
 - LLP2 – 0x0c0
 - LLP3 – 0x118
 - LLP4 – 0x170
 - LLP5 – 0x1c8
 - LLP6 – 0x220
 - LLP7 – 0x278
- **Read/write access:** read/write

This register does not exist if the DMAH_CHx_HC_LLDP configuration parameter is set to True.

Note: You need to program this register to point to the first Linked List Item (LLI) in memory prior to enabling the channel if block chaining is enabled – rows 6 to 10 of Table 9-19.

If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

Bit	Name	Access	Reset	Description
63:32	RSVD	N/A	0x0	Reserved
31:2	LOC	R/W	0x0	Starting Address in Memory of next LLI if block chaining is enabled. Note that the two LSBs of the starting address are not stored because the address is assumed to be aligned to a 32-bit boundary. LLI accesses are always 32-bit accesses (Hsize = 2) aligned to 32-bit boundaries and cannot be changed or programmed to anything other than 32-bit.
1:0	LMS	R/W	0x0	List Master Select. Identifies the AHB layer/interface where the memory device that stores the next linked list item resides. 00 = AHB master 1 01 = AHB master 2 10 = AHB master 3 11 = AHB master 4 This field does not exist if the configuration parameter DMAH_CHx_LMS is not set to NO_HARDCODE. In this case, the read-back value is always the hardcoded value. The maximum value of this field that can be read back is DMAH_NUM_MASTER_INT – 1.

The LLP register has two functions:

- The logical result of the equation $LLP.LOC \neq 0$ is used to set up the type of DMA transfer—single or multi-block. Table 9-19 shows how the method of updating the channel registers is a function of $LLP.LOC \neq 0$. If $LLP.LOC$ is set to 0x0, then transfers using linked lists are *not* enabled. This register must be programmed prior to enabling the channel in order to set up the transfertype.
- $LLP.LOC \neq 0$ contains the pointer to the next LLI for block chaining using linked lists. The LLPx register can also point to the address where write-back of the control and source/destination status information occur after block completion.

9.3.2.2.5 CTLx

- **Name:** Control Register for Channel x
- **Size:** 64 bits
- **Address offset:** for x = 0 to 7:
 - CTL0 – 0x018
 - CTL1 – 0x070
 - CTL2 – 0x0c8
 - CTL3 – 0x120
 - CTL4 – 0x178
 - CTL5 – 0x1d0
 - CTL6 – 0x228
 - CTL7 – 0x280
- **Read/write access:** read/write

This register contains fields that control the DMA transfer.

The CTLx register is part of the block descriptor (linked list item – LLI) when block chaining is enabled. It can be varied on a block-by-block basis within a DMA transfer when block chaining is enabled.

If status write-back is enabled, the upper word of the control register, CTLx[63:32], is written to the control register location of the LLI in system memory at the end of the block transfer.

Note: You need to program this register prior to enabling the channel.

Bit	Name	Access	Reset	Description
63:45	RSVD	N/A	0x0	Reserved
44	DONE	R/W	0x0	<p>Done bit</p> <p>If status write-back is enabled, the upper word of the control register, CTLx[63:32], is written to the control register location of the Linked List Item (LLI) in system memory at the end of the block transfer with the done bit set.</p> <p>Software can poll the LLI CTLx.DONE bit to see when a block transfer is complete. The LLI CTLx.DONE bit should be cleared when the linked lists are set up in memory prior to enabling the channel.</p> <p>LLI accesses are always 32-bit accesses (Hsize = 2) aligned to 32-bit boundaries and cannot be changed or programmed to anything other than 32-bit.</p>
b:32 (See description)	BLOCK_TS	R/W	0x2	<p>Block Transfer Size.</p> <p>When the DMAC is the flow controller, the user writes this field before the channel is enabled in order to indicate the block size. The number programmed into BLOCK_TS indicates the total number of single transactions to perform for every block transfer; a single transaction is mapped to a single AMBA beat.</p> <p>Width: The width of the single transaction is determined by CTLx.SRC_TR_WIDTH.</p> <p>Once the transfer starts, the read-back value is the total number of data items already read from the source peripheral, regardless of what is the flow controller.</p> <p>When the source or destination peripheral is assigned as the flow controller, then the maximum block size that can be read back saturates at DMAH_CHx_MAX_BLK_SIZE, but the actual block size can be greater.</p> <p>$b = \log_2(\text{DMAH_CHx_MAX_BLK_SIZE} + 1) + 31$</p> <p>Bit[43:b+1] do not exist and return 0 on a read.</p>
31:29	RSVD	N/A	0x0	Reserved

28	LLP_SRC_EN	R/W	0x0	Block chaining is enabled on the source side only if the LLP_SRC_EN field is high and LLPx.LOC is non-zero. Dependencies: This field does not exist if the configuration parameter DMAH_CHx_MULTI_BLK_EN is not selected or if DMAH_CHx_HC_LLP is selected; in this case, the read-back value is always 0.
27	LLP_DST_EN	R/W	0x0	Block chaining is enabled on the destination side only if the LLP_DST_EN field is high and LLPx.LOC is non-zero. Dependencies: This field does not exist if the configuration parameter DMAH_CHx_MULTI_BLK_EN is not selected or if DMAH_CHx_HC_LLP is selected; in this case, the read-back value is always 0.
26:25	SMS	R/W	DMAH_CHx_SMS[1:0]	Source Master Select. Identifies the Master Interface layer from which the source device (peripheral or memory) is accessed. 00 = AHB master 1 01 = AHB master 2 10 = AHB master 3 11 = AHB master 4 The maximum value of this field that can be read back is DMAH_NUM_MASTER_INT – 1. Dependencies: This field does not exist if the configuration parameter DMAH_CHx_SMS is hardcoded; in this case, the read-back value is always the hardcoded value.
24:23	DMS	R/W	DMAH_CHx_DMS[1:0]	Destination Master Select. Identifies the Master Interface layer from which the destination device (peripheral or memory) resides. 00 = AHB master 1 01 = AHB master 2 10 = AHB master 3 11 = AHB master 4 The maximum value of this field that can be read back is DMAH_NUM_MASTER_INT – 1. Dependencies: This field does not exist if the configuration parameter DMAH_CHx_DMS is hardcoded; in this case, the read-back value is always the hardcoded value.
22:20	TT_FC	R/W	See description	Transfer Type and Flow Control. The following transfer types are supported. Memory to Memory Memory to Peripheral Peripheral to Memory Peripheral to Peripheral Flow Control can be assigned to the DMAC, the source peripheral, or the destination peripheral. Table 9-17 lists the decoding for this field. Reset Value: Configuration dependent: TT_FC[0] = 1'b1 TT_FC[1] = DMAH_CHx_FC[1] (!DMAH_CHx_FC[0]) TT_FC[2] = DMAH_CHx_FC[1] ^ DMAH_CHx_FC[0] Dependencies: If the configuration parameter DMAH_CHx_FC (page 107) is set to DMA_FC_ONLY, then TT_FC[2] does not exist and TT_FC[2] always reads back 0. If DMAH_CHx_FC is set to SRC_FC_ONLY, then TT_FC[2:1] does not exist and TT_FC[2:1] always reads back 2'b10. If DMAH_CHx_FC is set to DST_FC_ONLY, then TT_FC[2:1] does not exist and TT_FC[2:1] always reads back 2'b11. For multi-block transfers using linked list operation, TT_FC must be constant for all blocks of this multi-block transfer.

19	RSVD	N/A	0x0	Reserved
18	DST_SCATTER_EN	R/W	0x0	<p>Destination scatter enable bit:</p> <p>0 = Scatter disabled</p> <p>1 = Scatter enabled</p> <p>Scatter on the destination side is applicable only when the CTLx.DINC bit indicates an incrementing or decrementing address control.</p> <p>Dependencies: This field does not exist if DMAH_CHx_DST_SCA_EN is not selected; in this case, the read-back value is always 0.</p>
17	SRC_GATHER_EN	R/W	0x0	<p>Source gather enable bit:</p> <p>0 = Gather disabled</p> <p>1 = Gather enabled</p> <p>Gather on the source side is applicable only when the CTLx.SINC bit indicates an incrementing or decrementing address control.</p> <p>Dependencies: This field does not exist if DMAH_CHx_SRC_GAT_EN is not selected; in this case, the read-back value is always 0.</p>
16:14	SRC_MSIZ	R/W	0x1	<p>Source Burst Transaction Length. Number of data items, each of width CTLx.SRC_TR_WIDTH, to be read from the source every time a source burst transaction request is made from the corresponding hardware handshaking interface. Table 9-15 lists the decoding for this field.</p> <p>Note: This value is not related to the AHB bus master HBURST bus.</p> <p>Dependencies: The configuration parameter DMAH_CHx_MAX_MULT_SIZE determines the bit width of this field. All remaining bits in this field do not exist and read back as 0.</p>
13:11	DEST_MSIZ	R/W	0x1	<p>Destination Burst Transaction Length. Number of data items, each of width CTLx.DST_TR_WIDTH, to be written to the destination every time a destination burst transaction request is made from the corresponding hardware handshaking interface. Table 9-15 lists the decoding for this field.</p> <p>Note: This value is not related to the AHB bus master HBURST bus.</p> <p>Dependencies: DMAH_CHx_MAX_MULT_SIZE determines the bit width of this field. All surplus bits in this field do not exist and read back as 0.</p>
10:9	SINC	R/W	0x0	<p>Source Address Increment. Indicates whether to increment or decrement the source address on every source transfer. If the device is fetching data from a source peripheral FIFO with a fixed address, then set this field to “No change.”</p> <p>00 = Increment</p> <p>01 = Decrement</p> <p>1x = No change</p> <p>Note: Incrementing or decrementing is done for alignment to the next CTLx.SRC_TR_WIDTH boundary.</p>
8:7	DINC	R/W	0x0	<p>Destination Address Increment. Indicates whether to increment or decrement the destination address on every destination transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to “No change.”</p> <p>00 = Increment</p> <p>01 = Decrement</p> <p>1x = No change</p>

				Note: Incrementing or decrementing is done for alignment to the next CTLx.DST_TR_WIDTH boundary.
6:4	SRC_TR_WIDTH	R/W	See description	Source Transfer Width. Table 9-16 lists the decoding for this field. Mapped to AHB bus "hsize." For a non-memory peripheral, typically the peripheral (source) FIFO width. This value must be less than or equal to DMAH_Mx_HDATA_WIDTH, where x is the AHB layer 1 to 4 where the source resides. Reset Value: Encoded value; refer to Table 9-16. Dependencies: This field does not exist if the parameter DMAH_CHx_STW is hardcoded. In this case, the read-back value is always the hardcoded source transfer width, DMAH_CHx_STW.
3:1	DST_TR_WIDTH	R/W	See description	Destination Transfer Width. Table 9-16 lists the decoding for this field. Mapped to AHB bus "hsize." For a non-memory peripheral, typically rgw peripheral (destination) FIFO width. This value must be less than or equal to DMAH_Mk_HDATA_WIDTH, where k is the AHB layer 1 to 4 where the destination resides. Reset Value: Encoded value; refer to Table 9-16. Dependencies: This field does not exist if DMAH_CHx_DTW is hardcoded. In this case, the read-back value is always the hardcoded destination transfer width, DMAH_CHx_DTW.
0	INT_EN	R/W	0x1	Interrupt Enable Bit. If set, then all interrupt-generating sources are enabled. Functions as a global mask bit for all interrupts for the channel; raw* interrupt registers still assert if CTLx.INT_EN = 0.

Table 9-15 CTLx.SRC_MSIZ and DEST_MSIZ decoding

CTLx.SRC_MSIZ/CTLx.DEST_MSIZ	Number of data items to be transferred (of width CTLx.SRC_TR_WIDTH or CTLx.DST_TR_WIDTH)
000	1
001	4
010	8
011	16
100	32
101	64
110	128
111	256

Table 9-16 CTLx.SRC_TR_WIDTH and CTLx.DST_TR_WIDTH decoding

CTLx.SRC_TR_WIDTH/ TLx.DST_TR_WIDTH	Size (bits)
000	8
001	16
010	32
011	64
100	128
101	256
11x	256

Table 9-17 CTLx.TT_FC field decoding

CTLx.TT_FC Field	Transfer Type	Flow Controller
000	Memory to Memory	DMAC
001	Memory to Peripheral	DMAC
010	Peripheral to Memory	DMAC
011	Peripheral to Peripheral	DMAC

100	Peripheral to Memory	Peripheral
101	Peripheral to Peripheral	Source Peripheral
110	Memory to Peripheral	Peripheral
111	Peripheral to Peripheral	Destination Peripheral

9.3.2.2.6 SSTATx

- **Name:** Source Status Register for Channel x
- **Size:** 64 bits (upper 32 bits are reserved)
- **Address offset:** for x = 0 to 7:
 - SSTAT0 – 0x020
 - SSTAT1 – 0x078
 - SSTAT2 – 0x0d0
 - SSTAT3 – 0x128
 - SSTAT4 – 0x180
 - SSTAT5 – 0x1d8
 - SSTAT6 – 0x230
 - SSTAT7 – 0x288
- **Read/write access:** read/write

After each block transfer completes, hardware can retrieve the source status information from the address pointed to by the contents of the SSTATARx register. This status information is then stored in the SSTATx register and written out to the SSTATx register location of the LLI before the start of the next block. This register does not exist if DMAH_CHx_STAT_SRC is set to False; in this case, the read-back value is always 0.

Note: This register is a temporary placeholder for the source status information on its way to the SSTATx register location of the LLI. The source status information should be retrieved by software from the SSTATx register location of the LLI, and not by a read of this register over the DMAC slave interface.

If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

Bit	Name	Access	Reset	Description
63:32	RSVD	N/A	0x0	Reserved
31:0	SSTAT	R/W	0x0	Source status information retrieved by hardware from the address pointed to by the contents of the SSTATARx register.

9.3.2.2.7 DSTATx

- **Name:** Destination Status Register for Channel x
- **Size:** 64 bits (upper 32 bits are reserved)
- **Address offset:** for x = 0 to 7:
 - DSTAT0 – 0x028
 - DSTAT1 – 0x080
 - DSTAT2 – 0x0d8
 - DSTAT3 – 0x130
 - DSTAT4 – 0x188
 - DSTAT5 – 0x1e0
 - DSTAT6 – 0x238
 - DSTAT7 – 0x290
- **Read/write access:** read/write

After the completion of each block transfer, hardware can retrieve the destination status information from the address pointed to by the contents of the DSTATARx register. This status information is then stored in the DSTATx register and written out to the DSTATx register location of the LLI before the start of the next block. This register does not exist if DMAH_CHx_STAT_DST is set to False; in this case, the read-back value is always 0.

Note: This register is a temporary placeholder for the destination status information on its way to the DSTATx register location of the LLI. The destination status information should be retrieved by software from the DSTATx register location of the LLI and not by a read of this register over the DMAC slave interface.

If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

Bit	Name	Access	Reset	Description
63:32	RSVD	N/A	0x0	Reserved
31:0	DSTAT	R/W	0x0	Destination status information retrieved by hardware from the address pointed to by the contents of the DSTATARx register.

9.3.2.2.8 SSTATARx

- **Name:** Source Status Address Register for Channel x
- **Size:** 64 bits (upper 32 bits are reserved)
- **Address offset:** for x = 0 to 7:
 - SSTATAR0 – 0x030
 - SSTATAR1 – 0x088
 - SSTATAR2 – 0x0e0
 - SSTATAR3 – 0x138
 - SSTATAR4 – 0x190
 - SSTATAR5 – 0x1e8
 - SSTATAR6 – 0x240
 - SSTATAR7 – 0x298
- **Read/write access:** read/write

After each block transfer completes, hardware can retrieve the source status information from the address pointed to by the contents of the SSTATARx register.

Note: If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

This register does not exist if the configuration parameter DMAH_CHx_STAT_SRC is set to False; in this case, the read-back value is always 0.

Bit	Name	Access	Reset	Description
63:32	RSVD	N/A	0x0	Reserved
31:0	SSTATAR	R/W	0x0	Pointer from where hardware can fetch the source status information, which is registered in the SSTATx register and written out to the SSTATx register location of the LLI before the start of the next block.

9.3.2.2.9 DSTATARx

- **Name:** Destination Status Address Register for Channel x
- **Size:** 64 bits (upper 32 bits are reserved)
- **Address offset:** for x = 0 to 7:
 - DSTATAR0 – 0x038
 - DSTATAR1 – 0x090
 - DSTATAR2 – 0x0e8
 - DSTATAR3 – 0x140
 - DSTATAR4 – 0x198
 - DSTATAR5 – 0x1f0
 - DSTATAR6 – 0x248
 - DSTATAR7 – 0x2a0
- **Read/write access:** read/write

After the completion of each block transfer, hardware can retrieve the destination status information from the address pointed to by the contents of the DSTATARx register.

Note: If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

This register does not exist if the configuration parameter DMAH_CHx_STAT_DST is set to False; in this case, the read-back value is always 0.

Bit	Name	Access	Reset	Description
63:32	RSVD	N/A	0x0	Reserved
31:0	DSTATAR	R/W	0x0	Pointer from where hardware can fetch the destination status information, which is registered in the DSTATx register and written out to the DSTATx register location of the LLI before the start of the next block.

9.3.2.2.10 CFGx

- **Name:** Configuration Register for Channel x
- **Size:** 64 bits (upper 32 bits are reserved)
- **Address offset:** for $x = 0$ to 7:
 - CFG0 – 0x040
 - CFG1 – 0x098
 - CFG2 – 0x0f0
 - CFG3 – 0x148
 - CFG4 – 0x1a0
 - CFG5 – 0x1f8
 - CFG6 – 0x250
 - CFG7 – 0x2a8
- **Read/write access:** read/write

This register contains fields that configure the DMA transfer. The channel configuration register remains fixed for all blocks of a multi-block transfer.

Note: You need to program this register prior to enabling the channel.

Bit	Name	Access	Reset	Description
63:47	RSVD	N/A	0x0	Reserved
b:43 (See notes)	DEST_PER	R/W	0x0	Assigns a hardware handshaking interface (0 - DMAH_NUM_HS_INT-1) to the destination of channel x if the CFGx.HS_SEL_DST field is 0; otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface through the assigned hardware handshaking interface. Note 1: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface. Note 2: This field does not exist if the configuration parameter DMAH_NUM_HS_INT is set to 0. Note 3: <ul style="list-style-type: none"> ● $b = 43$ if DMAH_NUM_HS_INT is 1 ● $b = \text{ceil}(\log_2(\text{DMAH_NUM_HS_INT})) + 42$ if DMAH_NUM_HS_INT is greater than 1 Bits 46:(b+1) do not exist and return 0 on a read.
b:39 (See notes)	SRC_PER	R/W	0x0	Assigns a hardware handshaking interface (0 - DMAH_NUM_HS_INT-1) to the source of channel x if the CFGx.HS_SEL_SRC field is 0; otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface through the assigned hardware handshaking interface.

				<p>Note 1: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.</p> <p>Note 2: This field does not exist if the configuration parameter DMAH_NUM_HS_INT is set to 0.</p> <p>Note 3:</p> <ul style="list-style-type: none"> ● b = 39 if DMAH_NUM_HS_INT is 1 ● b = $\text{ceil}(\log_2(\text{DMAH_NUM_HS_INT})) + 38$ if DMAH_NUM_HS_INT is greater than 1 <p>Bits 42:(b+1) do not exist and return 0 on a read.</p>
38	SS_UPD_EN	R/W	0x0	<p>Source Status Update Enable. Source status information is fetched only from the location pointed to by the SSTATARx register, stored in the SSTATx register and written out to the SSTATx location of the LLI if SS_UPD_EN is high.</p> <p>Note: This enable is applicable only if DMAH_CHx_STAT_SRC is set to True. This field does not exist if the configuration parameter DMAH_CHx_STAT_SRC is set to False; in this case, the read-back value is always 0.</p>
37	DS_UPD_EN	R/W	0x0	<p>Destination Status Update Enable. Destination status information is fetched only from the location pointed to by the DSTATARx register, stored in the DSTATx register and written out to the DSTATx location of the LLI if DS_UPD_EN is high.</p> <p>This field does not exist if the configuration parameter DMAH_CHx_STAT_DST is set to False; in this case, the read-back value is always 0.</p>
36:34	PROTCTL	R/W	0x1	<p>Protection Control bits used to drive the AHB HPROT[3:1] bus. The <i>AMBA Specification</i> recommends that the default value of HPROT indicates a non-cached, non-buffered, privileged data access. The reset value is used to indicate such an access.</p> <p>HPROT[0] is tied high because all transfers are data accesses, as there are no opcode fetches.</p> <p>There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals. Table 9-18 shows the mapping of bits in this field to the AHB HPROT[3:1] bus.</p>
33	FIFO_MODE	R/W	0x0	<p>FIFO Mode Select. Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced.</p> <p>0 = Space/data available for single AHB transfer of the specified transfer width.</p> <p>1 = Data available is greater than or equal to half the FIFO depth for destination transfers and space available is greater than half the fifo depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.</p>
32	FCMODE	R/W	0x0	<p>Flow Control Mode. Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.</p> <p>0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled.</p> <p>1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited so that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.</p>
31	RELOAD_DST	N/A	0x0	<p>Automatic Destination Reload. The DARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated.</p> <p>This register does not exist if the configuration parameter DMAH_CHx_MULTI_BLK_EN is not selected; in this case, the read-back value is always 0.</p>

30	RELOAD_SRC	R/W	0x0	Automatic Source Reload. The SARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated. This field does not exist if the configuration parameter DMAH_CHx_MULTI_BLK_EN is not selected; in this case, the read-back value is always 0.
29:20	MAX_ABRST	R/W	0x0	Maximum AMBA Burst Length. Maximum AMBA burst length that is used for DMA transfers on this channel. A value of 0 indicates that software is not limiting the maximum AMBA burst length for DMA transfers on this channel. This field does not exist if the configuration parameter DMAH_MABRST is not selected; in this case, the read-back value is always 0, and the maximum AMBA burst length cannot be limited by software.
19	SRC_HS_POL	R/W	0x0	Source Handshaking Interface Polarity. 0 = Active high 1 = Active low
18	DST_HS_POL	R/W	0x0	Destination Handshaking Interface Polarity. 0 = Active high 1 = Active low
17	LOCK_B	R/W	0x0	Bus Lock Bit. When active, the AHB bus master signal hlock is asserted for the duration specified in CFGx.LOCK_B_L. This field does not exist if the configuration parameter DMAH_CHx_LOCK_EN is set to False; in this case, the read-back value is always 0.
16	LOCK_CH	R/W	0x0	Channel Lock Bit. When the channel is granted control of the master bus interface and if the CFGx.LOCK_CH bit is asserted, then no other channels are granted control of the master bus interface for the duration specified in CFGx.LOCK_CH_L. Indicates to the master bus interface arbiter that this channel wants exclusive access to the master bus interface for the duration specified in CFGx.LOCK_CH_L. This field does not exist if the configuration parameter DMAH_CHx_LOCK_EN is set to False; in this case, the read-back value is always 0.
15:14	LOCK_B_L	R/W	0x0	Bus Lock Level. Indicates the duration over which CFGx.LOCK_B bit applies. 00 = Over complete DMA transfer 01 = Over complete DMA block transfer 1x = Over complete DMA transaction This field does not exist if the parameter DMAH_CHx_LOCK_EN is set to False; in this case, the read-back value is always 0.
13:12	LOCK_CH_L	R/W	0x0	Channel Lock Level. Indicates the duration over which CFGx.LOCK_CH bit applies. 00 = Over complete DMA transfer 01 = Over complete DMA block transfer 1x = Over complete DMA transaction This field does not exist if the configuration parameter DMAH_CHx_LOCK_EN is set to False; in this case, the read-back value is always 0.
11	HS_SEL_SRC	N/A	0x1	Source Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for source requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware-initiated transaction requests are ignored. If the source peripheral is memory, then this bit is ignored.
10	HS_SEL_DST	R/W	0x1	Destination Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for destination requests on this channel.

				0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware- initiated transaction requests are ignored. If the destination peripheral is memory, then this bit is ignored.
9	FIFO_EMPTY	R/W	0x1	Indicates if there is data left in the channel FIFO. Can be used in conjunction with CFGx.CH_SUSP to cleanly disable a channel. 1 = Channel FIFO empty 0 = Channel FIFO not empty
8	CH_SUSP	R/W	0x0	Channel Suspend. Suspends all DMA data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CFGx.FIFO_EMPTY to cleanly disable a channel without losing any data. 0 = Not suspended. 1 = Suspend DMA transfer from the source.
7:5	CH_PRIOR	R/W	Channel Number. For example: Chan0=0 Chan1=1	Channel priority. A priority of 7 is the highest priority, and 0 is the lowest. This field must be programmed within the following range: 0: (DMAH_NUM_CHANNELS – 1) A programmed value outside this range will cause erroneous behavior.
4:0	RSVD	N/A	0x0	Reserved

Table 9-18 PROTCTL field to HPROT mapping

1'b1	HPROT[0]
CFGx.PROTCTL[1]	HPROT[1]
CFGx.PROTCTL[2] ->	HPROT[2]
CFGx.PROTCTL[3] ->	HPROT[3]

9.3.2.2.11 SGRx

- **Name:** Source Gather Register for Channel x
- **Size:** 64 bits (upper 32 bits are reserved)
- **Address offset:** for x = 0 to 7:
 - SGR0 – 0x048
 - SGR1 – 0x0a0
 - SGR2 – 0x0f8
 - SGR3 – 0x150
 - SGR4 – 0x1a8
 - SGR5 – 0x200
 - SGR6 – 0x258
 - SGR7 – 0x2b0
- **Read/write access:** read/write

The Source Gather register contains two fields:

- Source gather count field (SGRx.SGC) – Specifies the number of contiguous source transfers of CTLx.SRC_TR_WIDTH between successive gather intervals. This is defined as a gather boundary.
- Source gather interval field (SGRx.SGI) – Specifies the source address increment/decrement in multiples of CTLx.SRC_TR_WIDTH on a gather boundary when gather mode is enabled for the source transfer.

Note: If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

The CTLx.SINC field controls whether the address increments or decrements. When the CTLx.SINC field indicates a fixed-address control, then the address remains constant throughout the transfer and the SGRx register is ignored. This register does not exist if the configuration parameter DMAH_CHx_SRC_GAT_EN is set to False.

Bit	Name	Access	Reset	Description
-----	------	--------	-------	-------------

63:32	RSVD	N/A	0x0	Reserved
$b:20$ (See description)	SGC	R/W	0x0	Source gather count. Source contiguous transfer count between successive gather boundaries. $b = \log_2 (\text{DMAH_CHx_MAX_BLK_SIZE} + 1) + 19$ Bit[31: $b+1$] do not exist and read back as 0.
19:0	SGL	R/W	0x0	Source gather interval.

9.3.2.2.12 DSRx

- **Name:** Destination Scatter Register for Channel x
- **Size:** 64 bits (upper 32 bits are reserved)
- **Address offset:** for x = 0 to 7:
 - DSR0 – 0x050
 - DSR1 – 0x0a8
 - DSR2 – 0x0100
 - DSR3 – 0x158
 - DSR4 – 0x1b0
 - DSR5 – 0x208
 - DSR6 – 0x260
 - DSR7 – 0x2b8
- **Read/write access:** read/write

The Destination Scatter register contains two fields:

- Destination scatter count field (DSRx.DSC) – Specifies the number of contiguous destination transfers of CTLx.DST_TR_WIDTH between successive scatter boundaries.
- Destination scatter interval field (DSRx.DSI) – Specifies the destination address increment/decrement in multiples of CTLx.DST_TR_WIDTH on a scatter boundary when scatter mode is enabled for the destination transfer.

Note: If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

The CTLx.DINC field controls whether the address increments or decrements. When the CTLx.DINC field indicates a fixed address control, then the address remains constant throughout the transfer and the DSRx register is ignored. This register does not exist if the configuration parameter DMAH_CHx_DST_SCA_EN is set to False.

Bit	Name	Access	Reset	Description
63:32	RSVD	N/A	0x0	Reserved
$b:20$ (See description)	DSC	R/W	0x0	Destination scatter count. Destination contiguous transfer count between successive scatter boundaries. $b = \log_2 (\text{DMAH_CHx_MAX_BLK_SIZE} + 1) + 19$ Bit[31: $b+1$] do not exist and read 0.
19:0	DSI	R/W	0x0	Destination scatter interval.

9.3.2.3 Interrupt Registers

The following sections describe the registers pertaining to interrupts, their status, and how to clear them. For each channel, there are five types of interrupt sources:

- IntBlock – Block Transfer Complete Interrupt
This interrupt is generated on DMA block transfer completion to the destination peripheral.
- IntDstTran – Destination Transaction Complete Interrupt
This interrupt is generated after completion of the last AHB transfer of the requested single/burst transaction from the handshaking interface on the destination side.
Note: If the destination for a channel is memory, then that channel will never generate the IntDstTran interrupt. Because of this, the corresponding bit in this field will not be set.
- IntErr – Error Interrupt

This interrupt is generated when an ERROR response is received from an AHB slave on the HRESP bus during a DMA transfer. In addition, the DMA transfer is cancelled and the channel is disabled.

- **IntSrcTran – Source Transaction Complete Interrupt**

This interrupt is generated after completion of the last AHB transfer of the requested single/burst transaction from the handshaking interface on the source side.

Note: If the source or destination is memory, then IntSrcTran/IntDstTran interrupts should be ignored, as there is no concept of a “DMA transaction level” for memory.

- **IntTfr – DMA Transfer Complete Interrupt**

This interrupt is generated on DMA transfer completion to the destination peripheral.

There are several groups of interrupt-related registers:

- RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr
- StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr
- MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr
- ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr
- StatusInt

When a channel has been enabled to generate interrupts, the following is true:

- Interrupt events are stored in the Raw Status registers.
- The contents of the Raw Status registers are masked with the contents of the Mask registers.
- The masked interrupts are stored in the Status registers.
- The contents of the Status registers are used to drive the int_* port signals.
- Writing to the appropriate bit in the Clear registers clears an interrupt in the Raw Status registers and the Status registers on the same clock cycle.

The contents of each of the five Status registers is ORed to produce a single bit for each interrupt type in the Combined Status register; that is, StatusInt.

Note: For interrupts to propagate past the raw* interrupt register stage, CTLX.INT_EN must be set to 1'b1, and the relevant interrupt must be unmasked in the mask* interrupt register.

9.3.2.3.1 RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr

- **Name:** Interrupt Raw Status Registers
- **Size:** 64 bits
- **Address offset:**
 - RawTfr – 0x2c0
 - RawBlock – 0x2c8
 - RawSrcTran – 0x2d0
 - RawDstTran – 0x2d8
 - RawErr – 0x2e0
- **Read/write access:** read/write

Interrupt events are stored in these Raw Interrupt Status registers before masking: RawBlock, RawDstTran, RawErr, RawSrcTran, and RawTfr. Each Raw Interrupt Status register has a bit allocated per channel; for example, RawTfr[2] is the Channel 2 raw transfer complete interrupt.

Each bit in these registers is cleared by writing a 1 to the corresponding location in the ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr registers.

Note: Write access is available to these registers for software testing purposes only. Under normal operation, writes to these registers are not recommended.

Bit	Name	Access	Reset	Description
63:DMAH_NUM_CHANNELS	RSVD	N/A	0x0	Reserved
DMAH_NUM_CHANNELS-1:0	RAW	R/W	0x0	Raw interrupt status.

9.3.2.3.2 StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr

- **Name:** Interrupt Status Registers
- **Size:** 64 bits
- **Address offset:**
 - StatusTfr – 0x2e8
 - StatusBlock – 0x2f0
 - StatusSrcTran – 0x2f8
 - StatusDstTran – 0x300
 - StatusErr – 0x308
- **Read/write access:** read

All interrupt events from all channels are stored in these Interrupt Status registers after masking: StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, and StatusTfr. Each Interrupt Status register has a bit allocated per channel; for example, StatusTfr[2] is the Channel 2 status transfer complete interrupt. The contents of these registers are used to generate the interrupt signals (int or int_n bus, depending on interrupt polarity) leaving the DMAC.

Bit	Name	Access	Reset	Description
63:DMAH_NUM_CHANNELS	RSVD	N/A	0x0	Reserved
DMAH_NUM_CHANNELS-1:0	STATUS	R	0x0	Interrupt status.

9.3.2.3.3 MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr

- **Name:** Interrupt Mask Registers
- **Size:** 64 bits
- **Address offset:**
 - MaskTfr – 0x310
 - MaskBlock – 0x318
 - MaskSrcTran – 0x320
 - MaskDstTran – 0x328
 - MaskErr – 0x330
- **Read/write access:** read/write

The contents of the Raw Status registers are masked with the contents of the Mask registers: MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, and MaskTfr. Each Interrupt Mask register has a bit allocated per channel; for example, MaskTfr[2] is the mask bit for the Channel 2 transfer complete interrupt.

When the source peripheral of DMA channel *i* is memory, then the source transaction complete interrupt, MaskSrcTran[*i*], must be masked to prevent an erroneous triggering of an interrupt on the int_combined signal. Similarly, when the destination peripheral of DMA channel *i* is memory, then the destination transaction complete interrupt, MaskDstTran[*i*], must be masked to prevent an erroneous triggering of an interrupt on the int_combined(_n) signal.

A channel INT_MASK bit will be written only if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same AHB write transfer. This allows software to set a mask bit without performing a read-modified write operation. For example, writing hex 01x1 to the MaskTfr register writes a 1 into MaskTfr[0], while MaskTfr[7:1] remains unchanged. Writing hex 00xx leaves MaskTfr[7:0] unchanged.

Writing a 1 to any bit in these registers unmask the corresponding interrupt, thus allowing the DMAC to set the appropriate bit in the Status registers and int_* port signals.

Bit	Name	Access	Reset	Description
63:8+dnc	RSVD	N/A	0x0	Reserved dnc = DMAH_NUM_CHANNELS
7+dnc:8	INT_MASK_WE	W	0x0	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled dnc = DMAH_NUM_CHANNELS
7:dnc	RSVD	N/A	0x0	Reserved

				dnc = DMAH_NUM_CHANNELS If dnc = 8, then this field is not present.
dnc-1:0	INT_MASK	R/W	0x0	Interrupt Mask 0 = masked 1 = unmasked dnc = DMAH_NUM_CHANNELS

9.3.2.3.4 ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr

- **Name:** Interrupt Clear Registers
- **Size:** 64 bits
- **Address offset:**
 - ClearTfr – 0x338
 - ClearBlock – 0x340
 - ClearSrcTran – 0x348
 - ClearDstTran – 0x350
 - ClearErr – 0x358
- **Read/write access:** write

Each bit in the Raw Status and Status registers is cleared on the same cycle by writing a 1 to the corresponding location in the Clear registers: ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, and ClearTfr. Each Interrupt Clear register has a bit allocated per channel; for example, ClearTfr[2] is the clear bit for the Channel 2 transfer complete interrupt. Writing a 0 has no effect. These registers are not readable.

Bit	Name	Access	Reset	Description
63:DMAH_NUM_CHANNELS	RSVD	N/A	0x0	Reserved
DMAH_NUM_CHANNELS-1:0	CLEAR	W	0x0	Interrupt clear. 0 = no effect 1 = clear interrupt

9.3.2.3.5 StatusInt

- **Name:** Combined Interrupt Status Register
- **Size:** 64 bits
- **Address offset:** 0x360
- **Read/write access:** read

The contents of each of the five Status registers – StatusTfr, StatusBlock, StatusSrcTran, StatusDstTran, StatusErr – is ORed to produce a single bit for each interrupt type in the Combined Status register (StatusInt). This register is read-only.

Bit	Name	Access	Reset	Description
63:5	RSVD	N/A	0x0	Reserved
4	ERR	R	0x0	OR of the contents of StatusErr register.
3	DSTT	R	0x0	OR of the contents of StatusDstTran register.
2	SRCT	R	0x0	OR of the contents of StatusSrcTran register.
1	BLOCK	R	0x0	OR of the contents of StatusBlock register.
0	TFR	R	0x0	OR of the contents of StatusTfr register.

9.3.2.4 Miscellaneous DMAC Registers

This section describes the following registers of the DMAC:

- DmaldReg – DMAC ID
- DmaTestReg – DMAC Test
- DMA Component ID Register – DMAC Version ID

9.3.2.4.1 DmaldReg

- **Name:** DMAC ID Register
- **Size:** 64 bits
- **Address offset:** 0x3a8
- **Read/write access:** read

This is the DMAC ID register, which is a read-only register that reads back the coreConsultant- configured hardcoded ID number, DMAH_ID_NUM.

Bit	Name	Access	Reset	Description
63:32	RSVD	N/A	0x0	Reserved
31:0	DMA_ID	R	DMA_ID_NUM	Hardcoded DMAC Peripheral ID

9.3.2.4.2 DmaTestReg

- **Name:** DMAC Test Register
- **Size:** 64 bits
- **Address offset:** 0x3b0
- **Read/write access:** read/write

This register is used to put the AHB slave interface into test mode, during which the readback value of the writable registers match the value written, assuming the DMAC configuration has not optimized the same registers. In normal operation, the readback value of some registers is a function of the DMAC state and does not match the value written.

Bit	Name	Access	Reset	Description
63:1	RSVD	N/A	0x0	Reserved
0	TEST_SLV_IF	R/W	0x0	Puts the AHB slave interface into test mode. In this mode, the read back value of the writable registers always matches the value written. This bit does not allow writing to read-only registers. 0 = Normal mode 1 = Test mode

9.3.2.4.3 DMA_COMP_PA RAMS_6

- **Name:** DMAC Component Parameters Register 6
- **Size:** 64 bits
- **Address offset:** 0x3c8
- **Read/write access:** read

This is a constant read-only register that contains encoded information about the component parameter settings for Channel 7. The reset value depends on coreConsultant parameter(s).

Note: If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

Bit	Name	Access	Description
63	RSVD	N/A	Reserved
62:60	CH7_FIFO_DEPTH	R	The value of this register is derived from the DMAH_CH7_FIFO_DEPTH coreConsultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 0x3 = 64 0x4 = 128

59:57	CH7_SMS	R	The value of this register is derived from the DMAH_CH7_SMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
56:54	CH7_LMS	R	The value of this register is derived from the DMAH_CH7_LMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
53:51	CH7_DMS	R	The value of this register is derived from the DMAH_CH7_DMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
50:48	CH7_MAX_MULT_SIZE	R	The value of this register is derived from the DMAH_CH7_MULT_SIZE coreConsultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
47:46	CH7_FC	R	The value of this register is derived from the DMAH_CH7_FC coreConsultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY
45	CH7_HC_LLP	R	The value of this register is derived from the DMAH_CH7_HC_LLP coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
44	CH7_CTL_WB_EN	R	The value of this register is derived from the DMAH_CH7_CTL_WB_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
43	CH7_MULTI_BLK_EN	R	The value of this register is derived from the DMAH_CH7_MULT_BLK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
42	CH7_LOCK_EN	R	The value of this register is derived from the DMAH_CH7_LOCK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
41	CH7_SRC_GAT_EN	R	The value of this register is derived from the DMAH_CH7_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
40	CH7_DST_SCA_EN	R	The value of this register is derived from the DMAH_CH7_DST_SCA_EN coreConsultant parameter.

			0x0 = FALSE 0x1 = TRUE
39	CH7_STAT_SRC	R	The value of this register is derived from the DMAH_CH7_STAT_SRC coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
38	CH7_STAT_DST	R	The value of this register is derived from the DMAH_CH7_STAT_DST coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
37:35	CH7_STW	R	The value of this register is derived from the DMAH_CH7_STW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
34:32	CH7_DTW	R	The value of this register is derived from the DMAH_CH7_DTW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
31:0	RSVD	N/A	Reserved

9.3.2.4.4 DMA_COMP_PA RAMS_5

- **Name:** DMAC Component Parameters Register 5
- **Size:** 64 bits
- **Address offset:** 0x3d0
- **Read/write access:** read

This is a constant read-only register that contains encoded information about the component parameter settings for Channel 5 and Channel 6. The reset value depends on coreConsultant parameter(s).

Note: If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

Bits 63:32 include the configuration parameters for Channel 5. Bits 31:0 include configuration parameters for Channel 6.

Bit	Name	Access	Description
63	RSVD	N/A	Reserved
62:60	CH5_FIFO_DEPTH	R	The value of this register is derived from the DMAH_CH5_FIFO_DEPTH coreConsultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 0x3 = 64 0x4 = 128

59:57	CH5_SMS	R	The value of this register is derived from the DMAH_CH5_SMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
56:54	CH5_LMS	R	The value of this register is derived from the DMAH_CH5_LMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
53:51	CH5_DMS	R	The value of this register is derived from the DMAH_CH5_DMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
50:48	CH5_MAX_MULT_SIZE	R	The value of this register is derived from the DMAH_CH5_MULT_SIZE coreConsultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
47:46	CH5_FC	R	The value of this register is derived from the DMAH_CH5_FC coreConsultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY
45	CH5_HC_LL	R	The value of this register is derived from the DMAH_CH5_HC_LL coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
44	CH5_CTL_WB_EN	R	The value of this register is derived from the DMAH_CH5_CTL_WB_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
43	CH5_MULTI_BLK_EN	R	The value of this register is derived from the DMAH_CH5_MULT_BLK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
42	CH5_LOCK_EN	R	The value of this register is derived from the DMAH_CH5_LOCK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
41	CH5_SRC_GAT_EN	R	The value of this register is derived from the DMAH_CH5_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
40	CH5_DST_SCA_EN	R	The value of this register is derived from the DMAH_CH5_DST_SCA_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE

39	CH5_STAT_SRC	R	The value of this register is derived from the DMAH_CH5_STAT_SRC coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
38	CH5_STAT_DST	R	The value of this register is derived from the DMAH_CH5_STAT_DST coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
37:35	CH5_STW	R	The value of this register is derived from the DMAH_CH5_STW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
34:32	CH5_DTW	R	The value of this register is derived from the DMAH_CH5_DTW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
31	RSVD	N/A	Reserved
30:28	CH6_FIFO_DEPTH	R	The value of this register is derived from the DMAH_CH6_FIFO_DEPTH coreConsultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 0x3 = 64 0x4 = 128
27:25	CH6_SMS	R	The value of this register is derived from the DMAH_CH6_SMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
24:22	CH6_LMS	R	The value of this register is derived from the DMAH_CH6_LMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
21:19	CH6_DMS	R	The value of this register is derived from the DMAH_CH6_DMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
18:16	CH6_MAX_MULT_SIZE	R	The value of this register is derived from the DMAH_CH6_MULT_SIZE coreConsultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32

			0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
15:14	CH6_FC	R	The value of this register is derived from the DMAH_CH6_FC coreConsultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY
13	CH6_HC_LLP	R	The value of this register is derived from the DMAH_CH6_HC_LLP coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
12	CH6_CTL_WB_EN	R	The value of this register is derived from the DMAH_CH6_CTL_WB_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
11	CH6_MULTI_BLK_EN	R	The value of this register is derived from the DMAH_CH6_MULT_BLK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
10	CH6_LOCK_EN	R	The value of this register is derived from the DMAH_CH6_LOCK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
9	CH6_SRC_GAT_EN	R	The value of this register is derived from the DMAH_CH6_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
8	CH6_DST_SCA_EN	R	The value of this register is derived from the DMAH_CH6_DST_SCA_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
7	CH6_STAT_SRC	R	The value of this register is derived from the DMAH_CH6_STAT_SRC coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
6	CH6_STAT_DST	R	The value of this register is derived from the DMAH_CH6_STAT_DST coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
5:3	CH6_STW	R	The value of this register is derived from the DMAH_CH6_STW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
2:0	CH6_DTW	R	The value of this register is derived from the DMAH_CH6_DTW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128

			0x6 = 256 0x7 = reserved
--	--	--	-----------------------------

9.3.2.4.5 DMA_COMP_PA RAMS_4

- **Name:** DMAC Component Parameters Register 4
- **Size:** 64 bits
- **Address offset:** 0x3d8
- **Read/write access:** read

This is a constant read-only register that contains encoded information about the component parameter settings for Channel 3 and Channel 4. The reset value depends on coreConsultant parameter(s).

Note: If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

Bits 63:32 include the configuration parameters for Channel 3. Bits 31:0 include configuration parameters for Channel 4.

Bit	Name	Access	Description
63	RSVD	N/A	Reserved
62:60	CH3_FIFO_DEPTH	R	The value of this register is derived from the DMAH_CH3_FIFO_DEPTH coreConsultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 0x3 = 64 0x4 = 128
59:57	CH3_SMS	R	The value of this register is derived from the DMAH_CH3_SMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
56:54	CH3_LMS	R	The value of this register is derived from the DMAH_CH3_LMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
53:51	CH3_DMS	R	The value of this register is derived from the DMAH_CH3_DMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
50:48	CH3_MAX_MULT_SIZE	R	The value of this register is derived from the DMAH_CH3_MULT_SIZE coreConsultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
47:46	CH3_FC	R	The value of this register is derived from the DMAH_CH3_FC coreConsultant parameter.

			0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY
45	CH3_HC_LLP	R	The value of this register is derived from the DMAH_CH3_HC_LLP coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
44	CH3_CTL_WB_EN	R	The value of this register is derived from the DMAH_CH3_CTL_WB_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
43	CH3_MULTI_BLK_EN	R	The value of this register is derived from the DMAH_CH3_MULT_BLK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
42	CH3_LOCK_EN	R	The value of this register is derived from the DMAH_CH3_LOCK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
41	CH3_SRC_GAT_EN	R	The value of this register is derived from the DMAH_CH3_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
40	CH3_DST_SCA_EN	R	The value of this register is derived from the DMAH_CH3_DST_SCA_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
39	CH3_STAT_SRC	R	The value of this register is derived from the DMAH_CH3_STAT_SRC coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
38	CH3_STAT_DST	R	The value of this register is derived from the DMAH_CH3_STAT_DST coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
37:35	CH3_STW	R	The value of this register is derived from the DMAH_CH3_STW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
34:32	CH3_DTW	R	The value of this register is derived from the DMAH_CH3_DTW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
31	RSVD	N/A	Reserved
30:28	CH4_FIFO_DEPTH	R	The value of this register is derived from the DMAH_CH4_FIFO_DEPTH coreConsultant parameter. 0x0 = 8

			0x1 = 16 0x2 = 32 0x3 = 64 0x4 = 128
27:25	CH4_SMS	R	The value of this register is derived from the DMAH_CH4_SMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
24:22	CH4_LMS	R	The value of this register is derived from the DMAH_CH4_LMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
21:19	CH4_DMS	R	The value of this register is derived from the DMAH_CH4_DMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
18:16	CH4_MAX_MULT_SIZE	R	The value of this register is derived from the DMAH_CH4_MULT_SIZE coreConsultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
15:14	CH4_FC	R	The value of this register is derived from the DMAH_CH4_FC coreConsultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY
13	CH4_HC_LLP	R	The value of this register is derived from the DMAH_CH4_HC_LLP coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
12	CH4_CTL_WB_EN	R	The value of this register is derived from the DMAH_CH4_CTL_WB_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
11	CH4_MULTI_BLK_EN	R	The value of this register is derived from the DMAH_CH4_MULT_BLK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
10	CH4_LOCK_EN	R	The value of this register is derived from the DMAH_CH4_LOCK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
9	CH4_SRC_GAT_EN	R	The value of this register is derived from the DMAH_CH4_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE

8	CH4_DST_SCA_EN	R	The value of this register is derived from the DMAH_CH4_DST_SCA_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
7	CH4_STAT_SRC	R	The value of this register is derived from the DMAH_CH4_STAT_SRC coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
6	CH4_STAT_DST	R	The value of this register is derived from the DMAH_CH4_STAT_DST coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
5:3	CH4_STW	R	The value of this register is derived from the DMAH_CH4_STW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
2:0	CH4_DTW	R	The value of this register is derived from the DMAH_CH4_DTW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved

9.3.2.4.6 DMA_COMP_PA RAMS_3

- **Name:** DMAC Component Parameters Register 3
- **Size:** 64 bits
- **Address offset:** 0x3e0
- **Read/write access:** read

This is a constant read-only register that contains encoded information about the component parameter settings for Channel 1 and Channel 2. The reset value depends on coreConsultant parameter(s).

Note: If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

Bits 63:32 include the configuration parameters for Channel 1. Bits 31:0 include configuration parameters for Channel 2.

Bit	Name	Access	Description
63	RSVD	N/A	Reserved
62:60	CH1_FIFO_DEPTH	R	The value of this register is derived from the DMAH_CH1_FIFO_DEPTH coreConsultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 0x3 = 64 0x4 = 128
59:57	CH1_SMS	R	The value of this register is derived from the DMAH_CH1_SMS coreConsultant parameter.

			0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
56:54	CH1_LMS	R	The value of this register is derived from the DMAH_CH1_LMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
53:51	CH1_DMS	R	The value of this register is derived from the DMAH_CH1_DMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
50:48	CH1_MAX_MULT_SIZE	R	The value of this register is derived from the DMAH_CH1_MULT_SIZE coreConsultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
47:46	CH1_FC	R	The value of this register is derived from the DMAH_CH1_FC coreConsultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY
45	CH1_HC_LL	R	The value of this register is derived from the DMAH_CH1_HC_LL coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
44	CH1_CTL_WB_EN	R	The value of this register is derived from the DMAH_CH1_CTL_WB_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
43	CH1_MULTI_BLK_EN	R	The value of this register is derived from the DMAH_CH1_MULT_BLK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
42	CH1_LOCK_EN	R	The value of this register is derived from the DMAH_CH1_LOCK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
41	CH1_SRC_GAT_EN	R	The value of this register is derived from the DMAH_CH1_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
40	CH1_DST_SCA_EN	R	The value of this register is derived from the DMAH_CH1_DST_SCA_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
39	CH1_STAT_SRC	R	The value of this register is derived from the DMAH_CH1_STAT_SRC coreConsultant parameter.

			0x0 = FALSE 0x1 = TRUE
38	CH1_STAT_DST	R	The value of this register is derived from the DMAH_CH1_STAT_DST coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
37:35	CH1_STW	R	The value of this register is derived from the DMAH_CH1_STW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
34:32	CH1_DTW	R	The value of this register is derived from the DMAH_CH1_DTW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
31	RSVD	N/A	Reserved
30:28	CH2_FIFO_DEPTH	R	The value of this register is derived from the DMAH_CH2_FIFO_DEPTH coreConsultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 0x3 = 64 0x4 = 128
27:25	CH2_SMS	R	The value of this register is derived from the DMAH_CH2_SMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
24:22	CH2_LMS	R	The value of this register is derived from the DMAH_CH2_LMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
21:19	CH2_DMS	R	The value of this register is derived from the DMAH_CH2_DMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
18:16	CH2_MAX_MULT_SIZE	R	The value of this register is derived from the DMAH_CH2_MULT_SIZE coreConsultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128

			0x6 = 256 0x7 = reserved
15:14	CH2_FC	R	The value of this register is derived from the DMAH_CH2_FC coreConsultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY
13	CH2_HC_LLP	R	The value of this register is derived from the DMAH_CH2_HC_LLP coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
12	CH2_CTL_WB_EN	R	The value of this register is derived from the DMAH_CH2_CTL_WB_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
11	CH2_MULTI_BLK_EN	R	The value of this register is derived from the DMAH_CH2_MULT_BLK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
10	CH2_LOCK_EN	R	The value of this register is derived from the DMAH_CH2_LOCK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
9	CH2_SRC_GAT_EN	R	The value of this register is derived from the DMAH_CH2_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
8	CH2_DST_SCA_EN	R	The value of this register is derived from the DMAH_CH2_DST_SCA_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
7	CH2_STAT_SRC	R	The value of this register is derived from the DMAH_CH2_STAT_SRC coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
6	CH2_STAT_DST	R	The value of this register is derived from the DMAH_CH2_STAT_DST coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
5:3	CH2_STW	R	The value of this register is derived from the DMAH_CH2_STW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
2:0	CH2_DTW	R	The value of this register is derived from the DMAH_CH2_DTW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved

9.3.2.4.7 DMA_COMP_PA RAMS_2

- **Name:** DMAC Component Parameters Register 2
- **Size:** 64 bits
- **Address offset:** 0x3e8
- **Read/write access:** read

This is a constant read-only register that contains encoded information about the component parameter settings. The reset value depends on coreConsultant parameter(s).

Note: If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

Bits 63:32 include the Multi-Block type parameter for Channel 0 to Channel 7. Bits 31:0 include the configuration parameters for Channel 0.

Bit	Name	Access	Description
63:60	CH7_MULTI_BLK_TYPE	R	The value of these bit fields are derived from the DMAH_CHx_MULTI_BLK_TYPE coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = CONT_RELOAD 0x2 = RELOAD_CONT 0x3 = RELOAD_RELOAD 0x4 = CONT_LL 0x5 = RELOAD_LL 0x6 = LLP_CONT 0x7 = LLP_RELOAD 0x8 = LLP_LL
59:56	CH7_MULTI_BLK_TYPE	R	
55:52	CH7_MULTI_BLK_TYPE	R	
51:48	CH7_MULTI_BLK_TYPE	R	
47:44	CH7_MULTI_BLK_TYPE	R	
43:40	CH7_MULTI_BLK_TYPE	R	
39:36	CH7_MULTI_BLK_TYPE	R	
35:32	CH7_MULTI_BLK_TYPE	R	
31	RSVD	N/A	Reserved
30:28	CH0_FIFO_DEPTH	R	The value of this register is derived from the DMAH_CH0_FIFO_DEPTH coreConsultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 0x3 = 64 0x4 = 128
27:25	CH0_SMS	R	The value of this register is derived from the DMAH_CH0_SMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
24:22	CH0_LMS	R	The value of this register is derived from the DMAH_CH0_LMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE
21:19	CH0_DMS	R	The value of this register is derived from the DMAH_CH0_DMS coreConsultant parameter. 0x0 = MASTER_1 0x1 = MASTER_2 0x2 = MASTER_3 0x3 = MASTER_4 0x4 = NO_HARDCODE

18:16	CH0_MAX_MULT_SIZE	R	The value of this register is derived from the DMAH_CH0_MULT_SIZE coreConsultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
15:14	CH0_FC	R	The value of this register is derived from the DMAH_CH0_FC coreConsultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY
13	CH0_HC_LLP	R	The value of this register is derived from the DMAH_CH0_HC_LLP coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
12	CH0_CTL_WB_EN	R	The value of this register is derived from the DMAH_CH0_CTL_WB_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
11	CH0_MULTI_BLK_EN	R	The value of this register is derived from the DMAH_CH0_MULT_BLK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
10	CH0_LOCK_EN	R	The value of this register is derived from the DMAH_CH0_LOCK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
9	CH0_SRC_GAT_EN	R	The value of this register is derived from the DMAH_CH0_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
8	CH0_DST_SCA_EN	R	The value of this register is derived from the DMAH_CH0_DST_SCA_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
7	CH0_STAT_SRC	R	The value of this register is derived from the DMAH_CH0_STAT_SRC coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
6	CH0_STAT_DST	R	The value of this register is derived from the DMAH_CH0_STAT_DST coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE
5:3	CH0_STW	R	The value of this register is derived from the DMAH_CH0_STW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved

2:0	CH0_DTW	R	The value of this register is derived from the DMAH_CH0_DTW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved
-----	---------	---	---

9.3.2.4.8 DMA_COMP_PA RAMS_1

- **Name:** DMAC Component Parameters Register 1
- **Size:** 64 bits
- **Address offset:** 0x3f0
- **Read/write access:** read

This is a constant read-only register that contains encoded information about the component parameter settings. The reset value depends on coreConsultant parameter(s).

Note: If DMAH_RETURN_ERR_RESP is set to True, the DMAC returns an ERROR response to an illegal register access, which includes accessing registers that have been removed during DMAC configuration. If DMAH_RETURN_ERR_RESP is set to False, DMAC always returns an OK response.

Bits 63:32 include the top-level parameter for DMAC. Bits 31:0 include the maximum block size parameters for Channel 0 to Channel 7.

Bit	Name	Access	Description
63:62	RSVD	N/A	Reserved
61	STATIC_ENDIAN_SELECT	R	The value of this register is derived from the DMAH_STATIC_ENDIAN_SELECT coreConsultant parameter. 0 = FALSE 0 = TRUE
60	ADD_ENCODED_PARAMS	R	The value of this register is derived from the DMAH_ADD_ENCODED_PARAMS coreConsultant parameter. 0 = FALSE 0 = TRUE
59:55	NUM_HS_INT	R	The value of this register is derived from the DMAH_NUM_HS_INT coreConsultant parameter. 0x00 = 0 to 0x10 = 16
54:53	M4_HDATA_WIDTH	R	The value of this register is derived from the DMAH_M4_HDATA_WIDTH coreConsultant parameter. 0x0 = 32 bits 0x1 = 64 bits 0x2 = 128 bits 0x3 = 256 bits
52:51	M3_HDATA_WIDTH	R	The value of this register is derived from the DMAH_M3_HDATA_WIDTH coreConsultant parameter. 0x0 = 32 bits 0x1 = 64 bits 0x2 = 128 bits 0x3 = 256 bits
50:49	M2_HDATA_WIDTH	R	The value of this register is derived from the DMAH_M2_HDATA_WIDTH coreConsultant parameter. 0x0 = 32 bits

			0x1 = 64 bits 0x2 = 128 bits 0x3 = 256 bits
48:47	M1_HDATA_WIDTH	R	The value of this register is derived from the DMAH_M1_HDATA_WIDTH coreConsultant parameter. 0x0 = 32 bits 0x1 = 64 bits 0x2 = 128 bits 0x3 = 256 bits
46:45	S_HDATA_WIDTH	R	The value of this register is derived from the DMAH_S_HDATA_WIDTH coreConsultant parameter. 0x0 = 32 bits 0x1 = 64 bits 0x2 = 128 bits 0x3 = 256 bits
44:43	NUM_MASTER_INT	R	The value of this register is derived from the DMAH_NUM_MASTER_INT coreConsultant parameter. 0x00 = 1 to 0x10 = 4
42:40	NUM_CHANNELS	R	The value of this register is derived from the DMAH_NUM_CHANNELS coreConsultant parameter. 0x00 = 1 to 0x10 = 8
39:36	RSVD	N/A	Reserved
35	MABRST	R	The value of this register is derived from the DMAH_MABRST coreConsultant parameter. 0 = FALSE 0 = TRUE
34:33	INTR_IO	R	The value of this register is derived from the DMAH_INTR_IO coreConsultant parameter. 0x0 = ALL 0x1 = TYPE 0x2 = COMBINED 0x3 = reserved
32	BIG_ENDIAN	R	The value of this register is derived from the DMAH_BIG_ENDIAN coreConsultant parameter. 0 = FALSE 0 = TRUE
31:28	CH7_MAX_BLK_SIZE	R	The values of these bit fields are derived from the DMAH_CHx_MAX_BLK_SIZE coreConsultant parameter. 0x0 = 3 0x1 = 7 0x2 = 15 0x3 = 31 0x4 = 63 0x5 = 127 0x6 = 255 0x7 = 511 0x8 = 1023 0x9 = 2047 0xa = 4095
27:24	CH6_MAX_BLK_SIZE	R	
23:20	CH5_MAX_BLK_SIZE	R	
19:16	CH4_MAX_BLK_SIZE	R	
15:12	CH3_MAX_BLK_SIZE	R	
11:8	CH2_MAX_BLK_SIZE	R	
7:4	CH1_MAX_BLK_SIZE	R	
3:0	CH0_MAX_BLK_SIZE	R	

9.3.2.4.9 DMA Component ID Register

- **Name:** DMA Component ID Register

- **Size:** 64 bits
- **Address Offset:** 0x3f8
- **Read/Write Access:** Read

This is the DMAC Component Version register, which is a read-only register that specifies the version of the packaged component in the upper 32 bits and the component type in the lower 32 bits.

Bit	Name	Access	Reset	Description
63:32	DMA_COMP_VERSION	R	See DMAC releases table in <i>AMBA 2 Release Notes</i>	Version of the component.
31:0	DMA_COMP_TYPE	R	0x44_57_11_10	Component Type number = 0x44_57_11_10. This assigned unique hex value is constant.

9.4 Programming the DMAC

The DMAC can be programmed through software registers or the DMAC low-level software driver. Software registers are described in “Registers”.

Note: There are references to both software and hardware parameters throughout this chapter. The software parameters are the field names in each register description table and are prefixed by the register name; for example, the Block Transfer Size field in the Control Register for Channel x is designated as “CTLx.BLOCK_TS.”

Shipped with the DMAC component is an address definition (memory map) C header file. This can be used when the DMAC is programmed in a C environment.

9.4.1 Register Access

All registers are aligned to a 64-bit boundary and are 64 bits wide. In general, the upper 32 bits of a register are reserved. A write to reserved bits within the register is ignored. A read from reserved bits in the register reads back 0. To avoid address aliasing, do one of the following:

- (1) The DMAC should not be allocated more than 1 KB of address space in the system memory map. If it is, then addresses selected above 1 KB from the base address are aliased to an address within the 1 KB space, and a transfer takes place involving this register.
- (2) Software should not attempt to access non-register locations when hsel is asserted.

Note: The hsel signal is asserted by the system decoder when the address on the bus is within the system address assigned for DMAC.

9.4.2 Illegal Register Access

An illegal access can be any of the following:

- (1) A AHB transfer of hsize greater than 64 is attempted.
- (2) The hsel signal is asserted, but the address does not decode to a valid address.
- (3) A write to the SARx, DARx, LLPx, CTLx, SSTATx, DSTATx, SSTATARx, DSTATARx, SGRx, or DSRx registers occurs when the channel is enabled.
- (4) A read from the ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr is attempted.
- (5) A write to the StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr is attempted.
- (6) A write to the StatusInt register is attempted.
- (7) A write to either the DmaldReg or DMA Component ID Register register is attempted.

The response to an illegal access is configured using the configuration parameter DMAH_RETURN_ERR_RESP. When DMAH_RETURN_ERR_RESP is set to True, an illegal access (read/write) returns an error response.

If DMAH_RETURN_ERR_RESP is set to False, an OKAY response is returned, a read returns back 0x0, and a write is ignored.

9.4.3 DMA Transfer Types

A DMA transfer may consist of single or multi-block transfers. On successive blocks of a multi-block transfer, the SARx/DARx register in the DMAC is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading
- Contiguous address between blocks

On successive blocks of a multi-block transfer, the CTLx register in the DMAC is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading

When block chaining, using Linked Lists is the multi-block method of choice. On successive blocks, the LLPx register in the DMAC is reprogrammed using block chaining with linked lists.

A block descriptor consists of six registers: SARx, DARx, LLPx, CTLx, SSTATx, and DSTATx. The first four registers, along with the CFGx register, are used by the DMAC to set up and describe the block transfer.

Note: The term Link List Item (LLI) and block descriptor are synonymous.

9.4.3.1 Multi-Block Transfers

Multi-block transfers are enabled by setting the DMAH_CHX_MULTI_BLK_EN configuration parameter to True.

Note: Multi-block transfers—in which the source and destination are swapped during the transfer—are not supported. In a multi-block transfer, the direction must not change for the duration of the transfer.

9.4.3.1.1 Block Chaining Using Linked Lists

To enable multi-block transfers using block chaining, you must set the configuration parameter DMAH_CHX_MULTI_BLK_EN to True and the DMAH_CHX_HC_LLP parameter to False.

In this case, the DMAC reprograms the channel registers prior to the start of each block by fetching the block descriptor for that block from system memory. This is known as an LLI update.

DMAC block chaining uses a Linked List Pointer register (LLPx) that stores the address in memory of the next linked list item. Each LLI contains the corresponding block descriptors:

- SARx
- DARx
- LLPx
- CTLx
- SSTATx
- DSTATx

To set up block chaining, you program a sequence of Linked Lists in memory.

LLI accesses are always 32-bit accesses (Hsize = 2) aligned to 32-bit boundaries and cannot be changed or programmed to anything other than 32-bit, even if the AHB master interface of the LLI supports more than a 32-bit data width.

The SARx, DARx, LLPx, and CTLx registers are fetched from system memory on an LLI update. If configuration parameter DMAH_CHX_CTL_WB_EN = True, then the updated contents of the CTLx, SSTATx, and DSTATx registers are written back to memory on block completion. Fig 9-45 and Fig 9-46 show how you use chained linked lists in memory to define multi-block transfers using block chaining.

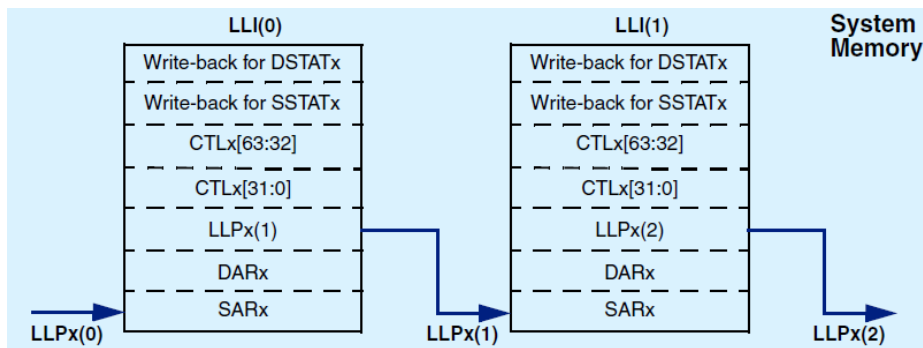


Fig 9-45 Multi-block transfer using linked lists when DMAH_CHx_STAT_SRC set to true

It is assumed that no allocation is made in system memory for the source status when the configuration parameter DMAH_CHx_STAT_SRC is set to False. If this parameter is False, then the order of a Linked List item is as follows:

- SARx
- DARx
- LLPx
- CTLx
- DSTATx

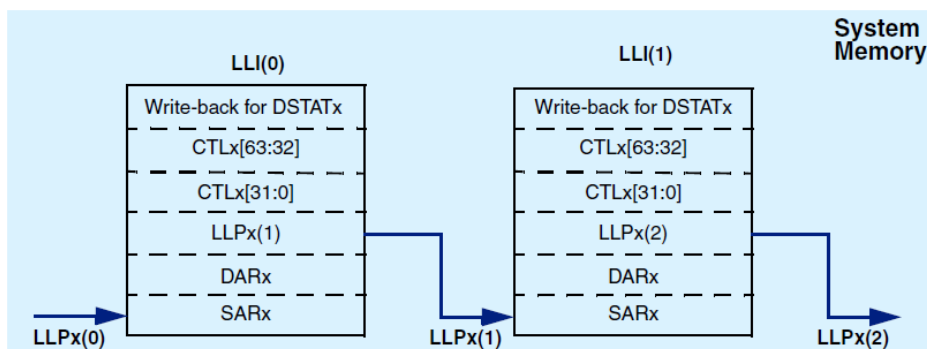


Fig 9-46 Multi-block transfer using linked lists when DMAH_CHx_STAT_SRC set to false

Note: In order to not confuse the SARx, DARx, LLPx, CTLx, STATx, and DSTATx register locations of the LLI with the corresponding DMAC memory mapped register locations, the LLI register locations are prefixed with LLI; that is, LLI.SARx, LLI.DARx, LLI.LLPx, LLI.CTLx, LLI.SSTATx, and LLI.DSTATx.

Fig 9-47 and Fig 9-48 show the mapping of a Linked List Item stored in memory to the channel registers block descriptor.

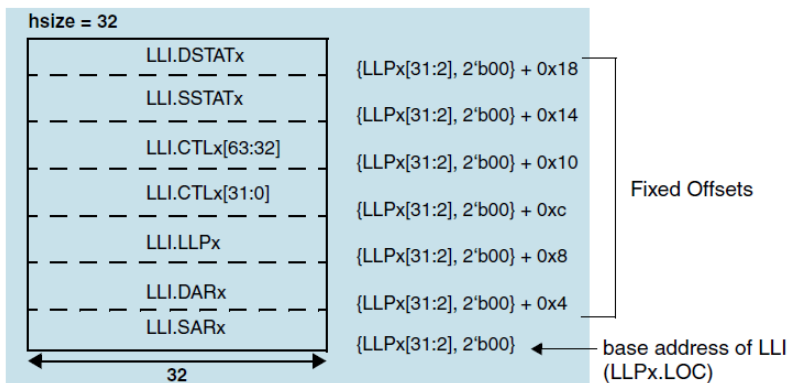


Fig 9-47 Mapping of block descriptor (LLI) in memory to channel registers when DMAH_CHx_STAT_SRC set to true

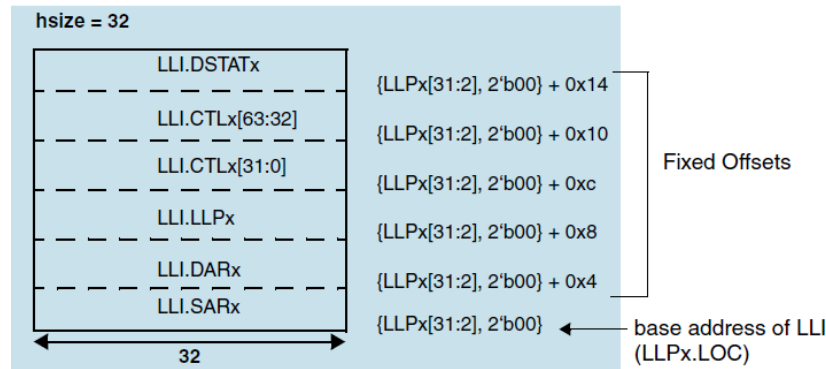


Fig 9-48 Mapping of block descriptor (LLI) in memory to channel registers when DMAH_CHx_STAT_SRC set to false

Rows 6 through 10 of Table 9-19 show the required values of LLPx, CTLx, and CFGx for multi-block DMA transfers using block chaining.

Note: For rows 6 through 10 of Table 9-19, the LLI.CTLx, LLI.LLPx, LLI.SARx, and LLI.DARx register locations of the LLI are always affected at the start of every block transfer. The LLI.LLPx and LLI.CTLx locations are always used to reprogram the DMAC LLPx and CTLx registers. However, depending on the Table 9-19 row number, the LLI.SARx/LLI.DARx address may or may not be used to reprogram the DMAC SARx/DARx registers.

Table 9-19 Programming of transfer types and channel register update method

No.	Transfer Type	LLP.LOC=0	LLP_SRC_EN (CTLx)	RELOAD_SRC (CRGx)	LLP_DST_EN (CTLx)	RELOAD_DST (CRGx)	CTLx, LLPx Update Method	SARx Update Method	DARx Update Method	Write Back ³
1	Single-block or last transfer of multi-block	Yes	0	0	0	0	None, user reprograms	None (Single)	None (Single)	No
2	Auto-reload multi-block transfer with contiguous SAR	Yes	0	0	0	1	CTLx, LLPx are reloaded from initial values	Contiguous	Auto-Reload	No
3	Auto-reload multi-block transfer with contiguous DAR	Yes	0	1	0	0	CTLx, LLPx are reloaded from initial values	Auto-Reload	Contiguous	No
4	Auto-reload multi-block transfer	Yes	0	1	0	1	CTLx, LLPx are reloaded from initial values	Auto-Reload	Auto-Reload	No
5	Single-block or last transfer of multi-block	No	0	0	0	0	None, user reprograms	None (Single)	None (Single)	Yes
6	Linked list multi-block transfer with contiguous SAR	No	0	0	1	0	CTLx, LLPx loaded from next Linked List item	Contiguous	Linked List	Yes
7	Linked list multi-block transfer with auto-reload SAR	No	0	1	1	0	CTLx, LLPx loaded from next Linked List item	Auto-Reload	Linked List	Yes
8	Linked list multi-block transfer with contiguous DAR	No	1	0	0	0	CTLx, LLPx loaded from next Linked List item	Linked List	Contiguous	Yes
9	Linked list multi-block transfer with auto-reload DAR	No	1	0	0	1	CTLx, LLPx loaded from next Linked List item	Linked List	Auto-Reload	Yes
10	Linked list multi-block transfer	No	1	0	1	0	CTLx, LLPx loaded from	Linked List	Linked List	Yes

³ This column assumes that the configuration parameter DMAH_CHx_CTL_WB_EN = True. If DMAH_CHx_CTL_WB_EN = False, then there is never writeback of the control and status registers regardless of transfer type, and all rows of this column are "No".

							next Linked List item			
--	--	--	--	--	--	--	--------------------------	--	--	--

Note:

- Throughout this databook, there are descriptions about fetching the LLI.CTLx register from the location pointed to by the LLPx register. This exact location is the LLI base address (stored in LLPx register) plus the fixed offset. For example, in Fig 9-47, the location of the LLI.CTLx register is LLPx.LOC + 0xc.
- Referring to Table 9-19, if the Write Back column entry is “Yes” and the configuration parameter DMAH_CHx_CTL_WB_EN = True, then the CTLx[63:32] register is always written to system memory (to LLI.CTLx[63:32]) at the end of every block transfer. The source status is fetched and written to system memory at the end of every block transfer if the Write Back column entry is “Yes,” DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and CFGx.SS_UPD_EN is enabled. The destination status is fetched and written to system memory at the end of every block transfer if the Write Back column entry is “Yes,” DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and CFGx.DS_UPD_EN is enabled.

9.4.3.2 Auto-Reloading of Channel Registers

During auto-reloading, the channel registers are reloaded with their initial values at the completion of each block and the new values used for the new block. Depending on the row number in Table 9-19, some or all of the SARx, DARx, and CTLx channel registers are reloaded from their initial value at the start of a block transfer.

9.4.3.3 Contiguous Address between Blocks

In this case, the address between successive blocks is selected as a continuation from the end of the previous block. Enabling the source or destination address to be contiguous between blocks is a function of the CTLx.LLP_SRC_EN, CFGx.RELOAD_SRC, CTLx.LLP_DST_EN, and CTLx.RELOAD_DST registers (see Table 9-19).

Note: You cannot select both SARx and DARx updates to be contiguous. If you want this functionality, you should increase the size of the Block Transfer (CTLx.BLOCK_TS), or if this is at the maximum value, use Row 10 of Table 9-19 and set up the LLI.SARx address of the block descriptor to be equal to the end SARx address of the previous block. Similarly, set up the LLI.DARx address of the block descriptor to be equal to the end DARx address of the previous block. For more information, refer to “Multi-Block Transfer with Linked List for Source and Linked List for Destination (Row 10)”

9.4.3.4 Suspension of Transfers between Blocks

At the end of every block transfer, an end-of-block interrupt is asserted if:

- (1) Interrupts are enabled, CTLx.INT_EN = 1, and
- (2) The channel block interrupt is unmasked, MaskBlock[n] = 1, where *n* is the channel number.

Note: The block-complete interrupt is generated at the completion of the block transfer to the destination.

For rows 6, 8, and 10 of Table 9-19, the DMA transfer does not stall between block transfers. For example, at the end-of-block *N*, the DMAC automatically proceeds to block *N* + 1.

For rows 2, 3, 4, 7, and 9 of Table 9-19 (SARx and/or DARx auto-reloaded between block transfers), the DMA transfer automatically stalls after the end-of-block interrupt is asserted, if the end-of-block interrupt is enabled and unmasked.

The DMAC does not proceed to the next block transfer until a write to the ClearBlock[n] block interrupt clear register, done by software to clear the channel block-complete interrupt, is detected by hardware.

For rows 2, 3, 4, 7, and 9 of Table 9-19 (SARx and/or DARx auto-reloaded between block transfers), the DMA transfer does not stall if either:

- Interrupts are disabled, CTLx.INT_EN = 0, or
- The channel block interrupt is masked, MaskBlock[n] = 0, where *n* is the channel number.

Channel suspension between blocks is used to ensure that the end-of-block ISR (interrupt service routine) of the next-to-last block is serviced before the start of the final block commences. This ensures that the ISR has cleared the CFGx.RELOAD_SRC and/or CFGx.RELOAD_DST bits before completion of the final block. The reload bits CFGx.RELOAD_SRC and/or CFGx.RELOAD_DST should be cleared in the end-of-block ISR for the next-to-last block transfer.

9.4.3.5 Ending Multi-Block Transfers

All multi-block transfers must end as shown in either Row 1 or Row 5 of Table 9-19. At the end of every block transfer, the DMAC samples the row number, and if the DMAC is in the Row 1 or Row 5 state, then the previous block transferred was the last block and the DMA transfer is terminated.

Note: Row 1 and Row 5 are used for single-block transfers or terminating multi-block transfers. Transfers initiated in rows 2, 3 or 4 can only end in row 1; similarly, transfers initiated in rows 6 through 10 can only end in row 5. Ending in the Row 5 state enables status fetch and write-back for the last block. Ending in the Row 1 state disables status fetch and write-back for the last block.

For rows 2, 3, and 4 of Table 9-19, (LLPx.LOC = 0 and CFGx.RELOAD_SRC and/or CFGx.RELOAD_DST is set), multi-block DMA transfers continue until both the CFGx.RELOAD_SRC and CFGx.RELOAD_DST registers are cleared by software. They should be programmed to 0 in the end-of-block interrupt service routine that services the next-to-last block transfer; this puts the DMAC into the Row 1 state.

For rows 6, 8, and 10 of Table 9-19 (both CFGx.RELOAD_SRC and CFGx.RELOAD_DST cleared), the user must set up the last block descriptor in memory so that both LLI.CTLx.LLP_SRC_EN and LLI.CTLx.LLP_DST_EN are 0.

The sampling of the LLPx.LOC bit takes place exclusively at the beginning of the transfer when the channel is enabled. This determines whether writeback is enabled throughout the complete transfer, and changing the value of this bit in subsequent blocks on the same transfer does not have any effect.

Note: The only allowed transitions between the rows of Table 9-19 are from any row into Row 1 or Row 5. As already stated, a transition into row 1 or row 5 is used to terminate the DMA transfer; all other transitions between rows are not allowed. Software must ensure that illegal transitions between rows do not occur between blocks of a multi-block transfer. For example, if block N is in row 10, then the only allowed rows for block N + 1 are rows 10 or 5.

9.4.4 Programing Example

The flow diagram in Fig 1-58 shows an overview of programming the DMA. This section explains the step-by-step programming of the DMAC. The example demonstrates row 10 of Table 9-19 for Multi-Block Transfer with Linked List for Source and Linked List for Destination. This example uses the DMAC to move four blocks of contiguous data from source to destination memory using the Linked List feature.

- (1) Set up the chain of Linked List items—otherwise known as block descriptors—in memory. Write the control information in the LLI.CTLx register location of the block descriptor for each LLI in memory for Channel 1. In the LLI.CTLx register, the following is programmed:
 - a) Set up the transfer type for a memory-to-memory transfer:
 - `ctlx[22:20] = 3'b000;`
 - b) Set up the transfer characteristics:
 - i. Transfer width for the source in the SRC_TR_WIDTH field
 - `ctlx[6:4] = 3'b001;`
 - ii. Transfer width for the destination in the DST_TR_WIDTH field
 - `ctlx[3:1] = 3'b001;`
 - iii. Source master layer in the SMS field where the source resides
 - `ctlx[26:25] = 2'b00;`
 - iv. Destination master layer in the DMS field where the destination resides
 - `ctlx[24:23] = 2'b00;`
 - v. Incrementing address for the source in the SINC field
 - `ctlx[10:9] = 2'b00;`
 - vi. Incrementing address for the destination in the DINC field
 - `ctlx[8:7] = 2'b00;`
- (2) Write the channel configuration information into the CFGx register for Channel 1:
 - a) HS_SEL_SRC/HS_SEL_DST bits select which of the handshaking interfaces—hardware or software—is active for source requests on this channel.
 - `cfgx[11] = 1'b0;`
 - `cfgx[10] = 1'b0;`

These settings are ignored because both the source and destination are memory types.
 - b) If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral by programming the SRC_PER and DEST_PER bits:
 - `cfgx[46:43] = 1'b0;`
 - `cfgx[42:39] = 1'b0;`

These settings are ignored because both the source and destination are memory types.

(3) The following For loop, shown as a programming example, sets the following:

- LLI.LLPx register locations of all LLI entries in memory (except the last) to non-zero and point to the base address of the next Linked List Item
- LLI.SARx/LLI.DARx register locations of all LLI entries in memory point to the start source/destination block address preceding that LLI fetch

The For statement below configures the LLPx entries:

```
for(i=0 ; i < 4 ; i=i+1) begin
    if (i == 3) llpx = 0; // end of LLI
    else llpx = llp_addr + 20; // start of next LLI

    //-: Program SAR
    `AHB_MASTER.write(0, llp_addr, sarx, AhbWord32Attrb, handle[0]);
    //-: Program DAR
    `AHB_MASTER.write(0, (llp_addr + 4), darx, AhbWord32Attrb, handle[0]);
    //-: Program LLP
    `AHB_MASTER.write(0, (llp_addr + 8), llpx, AhbWord32Attrb, handle[0]);
    //-: Program CTL
    `AHB_MASTER.write(0, (llp_addr + 12), ctlx[31:0], AhbWord32Attrb, handle[0]);
    `AHB_MASTER.write(0, (llp_addr + 16), ctlx[63:32], AhbWord32Attrb, handle[0]);
    // update pointers
    llp_addr = llp_addr + 20; // start of next LLI

    // 4 16-bit words each with scatter/gather interval in each block
    // ( will work only with scatter_gather count of 2)
    sarx = sarx + 24;
    darx = darx + 24;
end
```

- (4) If Gather is enabled—DMAH_CHx_SRC_GAT_EN = True and CTLx.SRC_GATHER_EN is enabled— program the SGRx register for Channel 1.
- (5) If Scatter is enabled—DMAH_CHx_DST_SCA_EN = True and CTLx.DST_SCATTER_EN is enabled—program the DSRx register for Channel 1.
- (6) Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers.
- (7) Finally, enable the channel by writing a 1 to the ChEnReg.CH_EN bit; the transfer is performed.

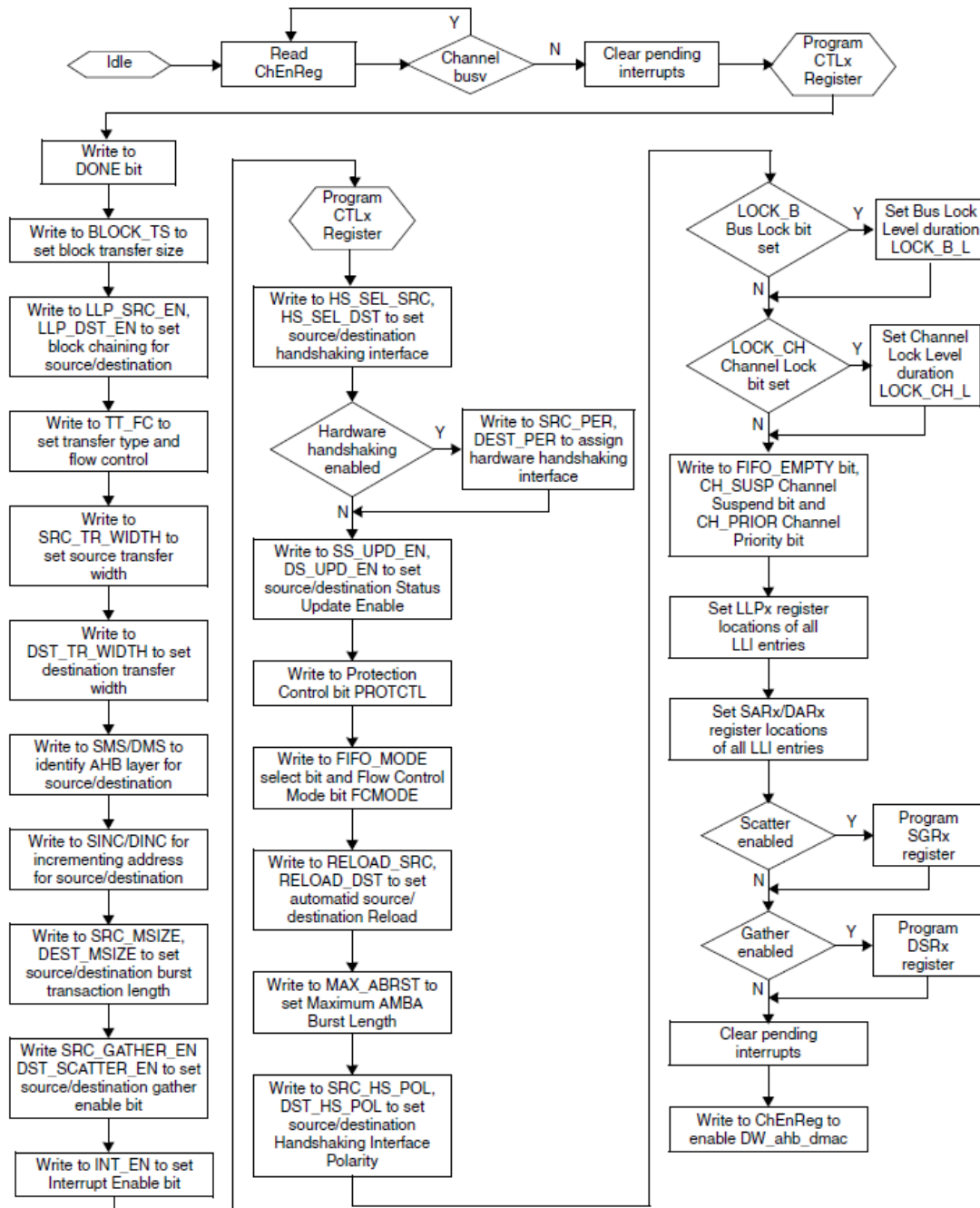


Fig 9-49 Flowchart for DMA programming example

9.4.5 Programming a Channel

Three registers – LLPx, CTLx, and CFGx – need to be programmed to determine whether single- or multi-block transfers occur, and which type of multi-block transfer is used. The different transfer types are shown in Table 9-19.

The DMAC can be programmed to fetch the status from the source or destination peripheral; this status is stored in the SSTATx and DSTATx registers. When the DMAC is programmed to fetch the status from the source or destination peripheral, it writes this status and the contents of the CTLx register back to memory at the end of a block transfer. The Write Back column of Table 9-19 shows when this occurs.

The “Update Method” columns indicate where the values of SARx, DARx, CTLx, and LLPx are obtained for the next block transfer when multi-block DMA transfers are enabled.

Note: In Table 9-19, all other combinations of LLPx.LOC = 0, CTLx.LLP_SRC_EN, CFGx.RELOAD_SRC, CTLx.LLP_DST_EN, and CFGx.RELOAD_DST are illegal, and will cause indeterminate or erroneous behavior.

The programming examples are as follows:

- Single-block Transfer (Row 1)
- Multi-Block Transfer with Linked List for Source and Linked List for Destination (Row 10)
- Multi-Block Transfer with Source Address Auto-Reloaded and DestinationAddress Auto-Reloaded (Row 4)
- Multi-Block Transfer with Source Address Auto-Reloaded and Linked List DestinationAddress (Row 7)
- Multi-Block Transfer with Source Address Auto-Reloaded and Contiguous DestinationAddress (Row 3)
- Multi-Block DMA Transfer with Linked List for Source and Contiguous DestinationAddress (Row 8)

9.4.5.1 Single-block Transfer (Row 1)

This section describes a single-block transfer, Row 1 in Table 9-19.

Note: Row 5 in Table 9-19 is also a single-block transfer with write-back of control and status information enabled at the end of the single-block transfer.

- (1) Read the Channel Enable register to choose a free (disabled) channel.
- (2) Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, and ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
- (3) Program the following channel registers:
 - a) Write the starting source address in the SARx register for channel x.
 - b) Write the starting destination address in the DARx register for channel x.
 - c) Program CTLx and CFGx according to Row 1, as shown in Table 9-19. Program the LLPx register with 0.
 - d) Write the control information for the DMA transfer in the CTLx register for channel x. For example, in the register, you can program the following:
 - i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register. Table 9-17 lists the decoding for this field.
 - ii. Set up the transfer characteristics, such as:
 - Transfer width for the source in the SRC_TR_WIDTH field. Table 9-16 lists the decoding for this field.
 - Transfer width for the destination in the DST_TR_WIDTH field. Table 9-16 lists the decoding for this field.
 - Source master layer in the SMS field where the source resides.
 - Destination master layer in the DMS field where the destination resides.
 - Incrementing/decrementing or fixed address for the source in the SINC field.
 - Incrementing/decrementing or fixed address for the destination in the DINC field.
 - e) Write the channel configuration information into the CFGx register for channel x.
 - i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.

This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests.

Writing a 1 activates the software handshaking interface to handle source and destination requests.
 - ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign a handshaking interface to the source and destination peripheral; this requires programming the SRC_PER and DEST_PER bits, respectively.
 - f) If gather is enabled (parameter DMAH_CHx_SRC_GAT_EN = True and CTLx.SRC_GATHER_EN is enabled), program the SGRx register for channel x.
 - g) If scatter is enabled (parameter DMAH_CHx_DST_SCA_EN = True and CTLx.DST_SCATTER_EN), program the DSRx register for channel x.
- (4) After the DMAC-selected channel has been programmed, enable the channel by writing a 1 to the ChEnReg.CH_EN bit. Ensure that bit 0 of the DmaCfgReg register is enabled.

- (5) Source and destination request single and burst DMA transactions in order to transfer the block of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
- (6) Once the transfer completes, hardware sets the interrupts and disables the channel. At this time, you can respond to either the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (RawTfr[n], n = channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, the software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, ClearTfr[n], before the channel is enabled.

9.4.5.2 Multi-Block Transfer with Linked List for Source and Linked List for Destination (Row 10)

Note: This type of multi-block transfer can only be enabled when either of the following parameters is set:

- DMAH_CHx_MULTI_BLK_TYPE = NO_HARDCODE
- DMAH_CHx_MULTI_BLK_TYPE = LLP_LLP

- (1) Read the Channel Enable register to choose a free (disabled) channel.
- (2) Set up the chain of Linked List Items (otherwise known as block descriptors) in memory. Write the control information in the LLI.CTLx register location of the block descriptor for each LLI in memory (see Fig 9-45) for channel x. For example, in the register, you can program the following:
 - a) Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register. Table 9-17 lists the decoding for this field.
 - b) Set up the transfer characteristics, such as:
 - Transfer width for the source in the SRC_TR_WIDTH field. Table 9-16 lists the decoding for this field.
 - Transfer width for the destination in the DST_TR_WIDTH field. Table 9-16 lists the decoding for this field.
 - Source master layer in the SMS field where the source resides.
 - Destination master layer in the DMS field where the destination resides.
 - Incrementing/decrementing or fixed address for the source in the SINC field.
 - Incrementing/decrementing or fixed address for the destination in the DINC field.
- (3) Write the channel configuration information into the CFGx register for channel x.
 - a) Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.
 - b) If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.
- (4) Make sure that the LLI.CTLx register locations of all LLI entries in memory (except the last) are set as shown in Row 10 of Table 9-19. The LLI.CTLx register of the last Linked List Item must be set as described in Row 1 or Row 5 of Table 9-19. Fig 9-45 shows a Linked List example with two list items.
- (5) Make sure that the LLI.LLPx register locations of all LLI entries in memory (except the last) are non-zero and point to the base address of the next Linked List Item.
- (6) Make sure that the LLI.SARx/LLI.DARx register locations of all LLI entries in memory point to the start source/destination block address preceding that LLI fetch.
- (7) If parameter DMAH_CHx_CTL_WB_EN = True, ensure that the LLI.CTLx.DONE field of the LLI.CTLx register locations of all LLI entries in memory is cleared.
- (8) If source status fetching is enabled (DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and CFGx.SS_UPD_EN is enabled), program the SSTATARx register so that the source status information can be fetched from the location pointed to by the SSTATARx. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of Table 9-19.
- (9) If destination status fetching is enabled (DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and CFGx.DS_UPD_EN is enabled), program the DSTATARx register so that the destination status information can be fetched from the location pointed to by the DSTATARx register. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of Table 9-19.
- (10) If gather is enabled (DMAH_CHx_SRC_GAT_EN = True and CTLx.SRC_GATHER_EN is enabled), program the SGRx register for channel x.
- (11) If scatter is enabled (DMAH_CHx_DST_SCA_EN = True and CTLx.DST_SCATTER_EN is enabled) program the DSRx register for channel x.
- (12) Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, and ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
- (13) Program the CTLx and CFGx registers according to Row 10, as shown in Table 9-19.
- (14) Program the LLPx register with LLP(0), the pointer to the first linked list item.

- (15) Finally, enable the channel by writing a 1 to the ChEnReg.CH_EN bit; the transfer is performed.
- (16) The DMAC fetches the first LLI from the location pointed to by LLPx(0).
Note: The LLI.SARx, LLI.DARx, LLI.LLPx, and LLI.CTLx registers are fetched. The DMAC automatically reprograms the SARx, DARx, LLPx, and CTLx channel registers from the LLPx(0).
- (17) Source and destination request single and burst DMA transactions to transfer the block of data (assuming non-memory peripheral). The DMAC acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
- (18) Once the block of data is transferred, the source status information is fetched from the location pointed to by the SSTATARx register and stored in the SSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and CFGx.SS_UPD_EN is enabled. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of Table 9-19. The destination status information is fetched from the location pointed to by the DSTATARx register and stored in the DSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and CFGx.DS_UPD_EN is enabled. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of Table 9-19.
- (19) If DMAH_CHx_CTL_WB_EN = True, then the CTLx[63:32] register is written out to system memory. For conditions under which the CTLx[63:32] register is written out to system memory, refer to the Write Back column of Table 9-19. The CTLx[63:32] register is written out to the same location on the same layer (LLPx.LMS) where it was originally fetched; that is, the location of the CTLx register of the linked list item fetched prior to the start of the block transfer. Only the second word of the CTLx register is written out – CTLx[63:32] – because only the CTLx.BLOCK_TS and CTLx.DONE fields have been updated by the DMAC hardware. Additionally, the CTLx.DONE bit is asserted to indicate block completion. Therefore, software can poll the LLI.CTLx.DONE bit of the CTLx register in the LLI to ascertain when a block transfer has completed.
Note: Do not poll the CTLx.DONE bit in the DMAC memory map; instead, poll the LLI.CTLx.DONE bit in the LLI for that block. If the polled LLI.CTLx.DONE bit is asserted, then this block transfer has completed. This LLI.CTLx.DONE bit was cleared at the start of the transfer (Step 7).
- (20) The SSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and CFGx.SS_UPD_EN is enabled. It is written to the SSTATx register location of the LLI pointed to by the previously saved LLPx.LOC register. The DSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and CFGx.DS_UPD_EN is enabled. It is written to the DSTATx register location of the LLI pointed to by the previously saved LLPx.LOC register. The end-of-block interrupt, int_block, is generated after the write-back of the control and status registers has completed.
Note: The write-back location for the control and status registers is the LLI pointed to by the previous value of the LLPx.LOC register, not the LLI pointed to by the current value of the LLPx.LOC register.
- (21) The DMAC does not wait for the block interrupt to be cleared, but continues fetching the next LLI from the memory location pointed to by the current LLPx register and automatically reprograms the SARx, DARx, LLPx, and CTLx channel registers. The DMA transfer continues until the DMAC determines that the CTLx and LLPx registers at the end of a block transfer match the ones described in Row 1 or Row 5 of Table 9-19 (as discussed earlier). The DMAC then knows that the previously transferred block was the last block in the DMA transfer.

The DMA transfer might look like that shown in Fig 9-50.

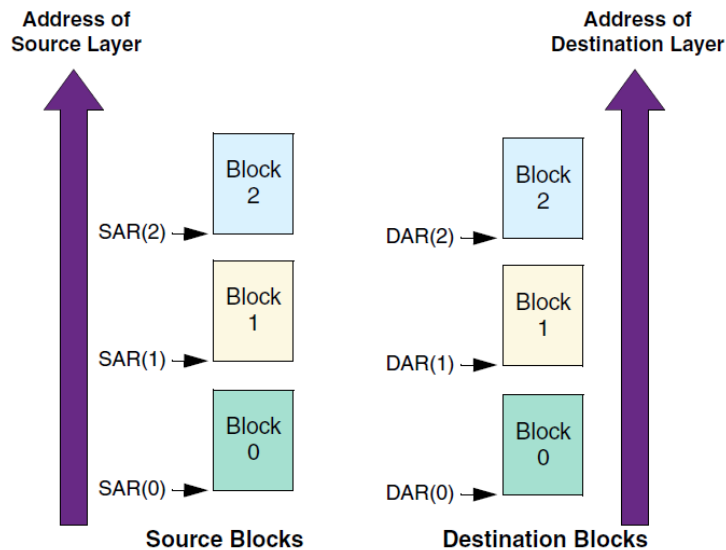


Fig 9-50 Multi-block with linked address for source and destination

If the user needs to execute a DMA transfer where the source and destination address are contiguous, but where the amount of data to be transferred is greater than the maximum block size CTLx.BLOCK_TS, then this can be achieved using the type of multi-block transfer shown in Fig 9-51.

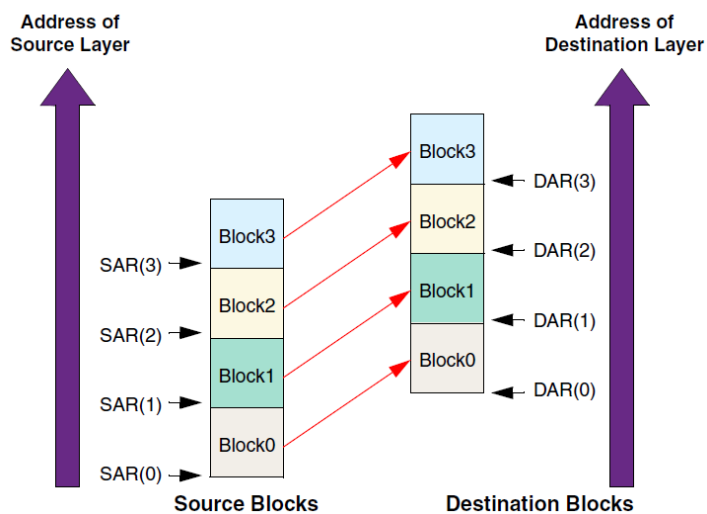


Fig 9-51 Multi-block with linked address for source and destination where SARx and DARx between successive blocks are contiguous

The DMA transfer flow is shown in Fig 9-52.

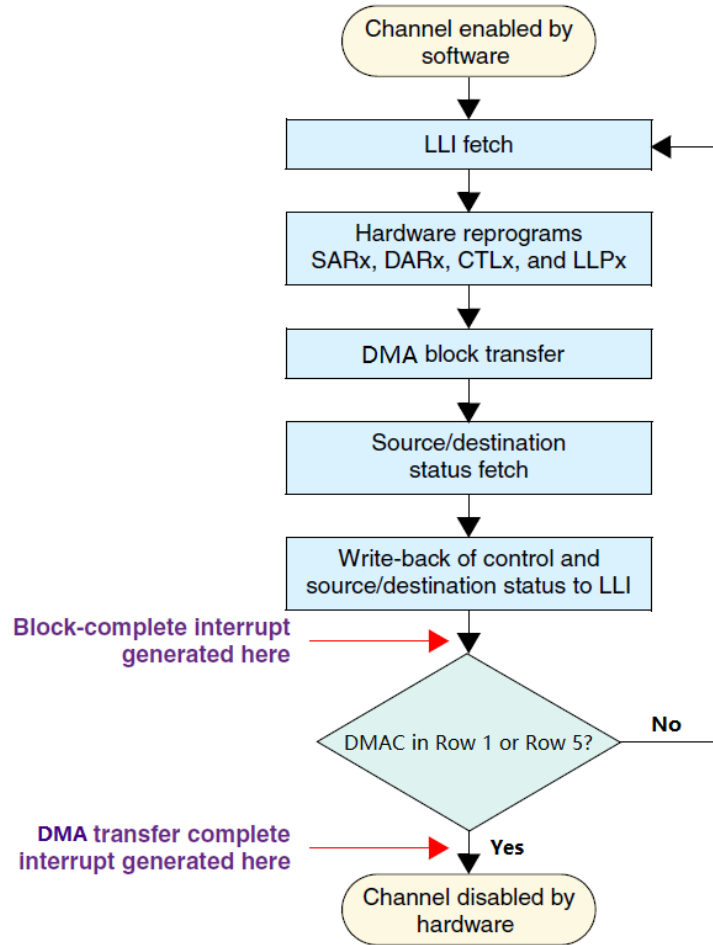


Fig 9-52 DMA transfer flow for source and destination linked list address

9.4.5.3 Multi-Block Transfer with Source Address Auto-Reloaded and Destination Address Auto-Reloaded (Row 4)

Note: This type of multi-block transfer can only be enabled when either of the following parameters is set:

- DMAH_CHx_MULTI_BLK_TYPE = NO_HARDCODE
- DMAH_CHx_MULTI_BLK_TYPE = RELOAD_RELOAD

- (1) Read the Channel Enable register to choose an available(disabled) channel.
- (2) Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, and ClearErr. Reading the Interrupt Raw Status and Interrupt Status register confirms that all interrupts have been cleared.
- (3) Program the following channel registers:
 - a) Write the starting source address in the SARx register for channelx.
 - b) Write the starting destination address in the DARx register for channelx.
 - c) Program CTLx and CFGx according to Row 4, as shown in Table 9-19. Program the LLPx register with 0.
 - d) Write the control information for the DMA transfer in the CTLx register for channel x. For example, in the register, you can program the following:
 - i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register. Table 9-17 lists the decoding for this field.
 - ii. Set up the transfer characteristics, such as:
 - Transfer width for the source in the SRC_TR_WIDTH field; Table 9-16 lists the decoding for this field.
 - Transfer width for the destination in the DST_TR_WIDTH field; Table 9-16 lists the decoding for this field.
 - Source master layer in the SMS field where the source resides.

- Destination master layer in the DMS field where the destination resides.
 - Incrementing/decrementing or fixed address for the source in the SINC field.
 - Incrementing/decrementing or fixed address for the destination in the DINC field.
- e) If gather is enabled ($\text{DMAH_CHx_SRC_GAT_EN} = \text{True}$ and $\text{CTLx.SRC_GATHER_EN}$ is enabled), program the SGRx register for channel x .
 - f) If scatter is enabled ($\text{DMAH_CHx_DST_SCA_EN} = \text{True}$ and $\text{CTLx.DST_SCATTER_EN}$), program the DSRx register for channel x .
 - g) Write the channel configuration information into the CFGx register for channel x . Ensure that the reload bits, CFGx.RELOAD_SRC and CFGx.RELOAD_DST , are enabled.
 - i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the $\text{HS_SEL_SRC/HS_SEL_DST}$ bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.
 - ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.
- (4) After the DMAC selected channel has been programmed, enable the channel by writing a 1 to the ChEnReg.CH_EN bit. Ensure that bit 0 of the DmaCfgReg register is enabled.
 - (5) Source and destination request single and burst DMAC transactions to transfer the block of data (assuming non-memory peripherals). The DMAC acknowledges on completion of each burst/single transaction and carries out the block transfer.
 - (6) When the block transfer has completed, the DMAC reloads the SARx , DARx , and CTLx registers. Hardware sets the block-complete interrupt. The DMAC then samples the row number, as shown in Table 9-19. If the DMAC is in Row 1, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register ($\text{RawTfr}[n]$, where n is the channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, $\text{ClearTfr}[n]$, before the channel is enabled. If the DMAC is not in Row 1, the next step is performed.
 - (7) The DMA transfer proceeds as follows:
 - a) If interrupts are enabled ($\text{CTLxx.INT_EN} = 1$) and the block-complete interrupt is unmasked ($\text{MaskBlock}[x] = 1'b1$, where x is the channel number), hardware sets the block-complete interrupt when the block transfer has completed. It then stalls until the block-complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block-complete ISR (interrupt service routine) should clear the reload bits in the CFGx.RELOAD_SRC and CFGx.RELOAD_DST registers. This puts the DMAC into Row 1, as shown in Table 9-19. If the next block is not the last block in the DMA transfer, then the reload bits should remain enabled to keep the DMAC in Row 4.
 - b) If interrupts are disabled ($\text{CTLx.INT_EN} = 0$) or the block-complete interrupt is masked ($\text{MaskBlock}[x] = 1'b0$, where x is the channel number), then hardware does not stall until it detects a write to the block-complete interrupt clear register; instead, it immediately starts the next block transfer. In this case, software must clear the reload bits in the CFGx.RELOAD_SRC and CFGx.RELOAD_DST registers to put the DMAC into Row 1 of Table 9-19 before the last block of the DMA transfer has completed.

The transfer is similar to that shown in Fig 9-53.

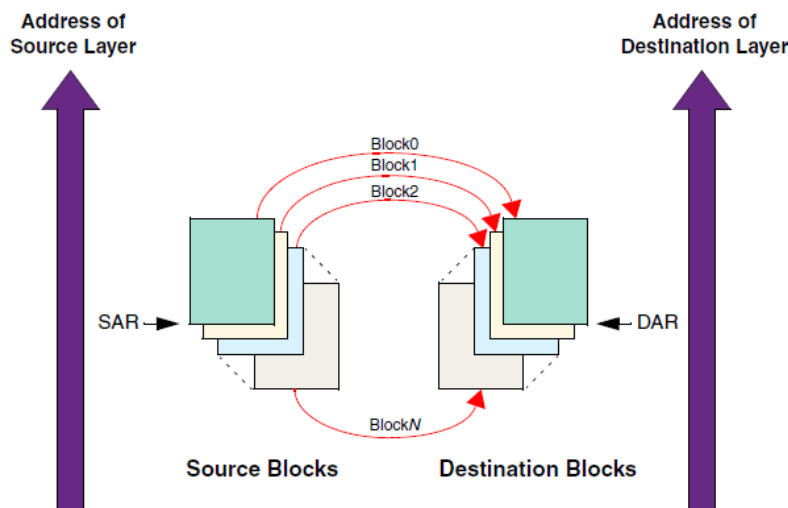


Fig 9-53 Multi-block DMA transfer with source and destination address auto-reloaded

The DMA transfer flow is shown in Fig 9-54.

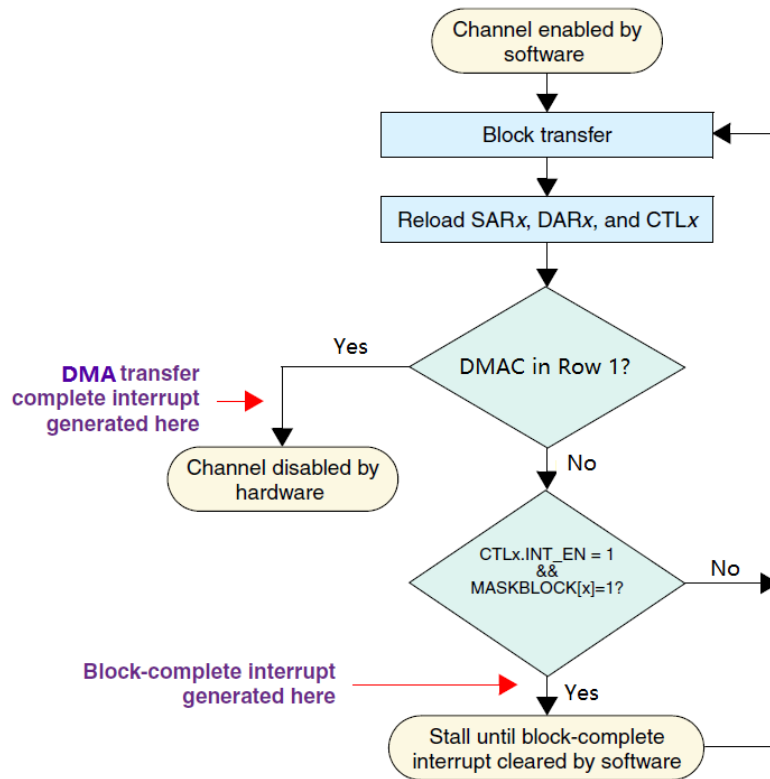


Fig 9-54 DMA transfer flow for source and destination address auto-reloaded

9.4.5.4 Multi-Block Transfer with Source Address Auto-Reloaded and Linked List Destination Address (Row 7)

Note: This type of multi-block transfer can only be enabled when either of the following parameters is set:

- DMAH_CHx_MULTI_BLK_TYPE = 0
- DMAH_CHx_MULTI_BLK_TYPE = RELOAD_LL

- (1) Read the Channel Enable register in order to choose a free (disabled) channel.
- (2) Set up the chain of Linked List Items (otherwise known as block descriptors) in memory. Write the control information in the LLI.CTLx register location of the block descriptor for each LLI in memory (see Fig 9-45) for channel x. For example, in the register, you can program the following:
 - a) Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register. Table 9-17 lists the decoding for this field.
 - b) Set up the transfer characteristics, such as:
 - i. Transfer width for the source in the SRC_TR_WIDTH field. Table 9-16 lists the decoding for this field.
 - ii. Transfer width for the destination in the DST_TR_WIDTH field. Table 9-16 lists the decoding for this field.
 - iii. Source master layer in the SMS field where the source resides.
 - iv. Destination master layer in the DMS field where the destination resides.
 - v. Incrementing/decrementing or fixed address for the source in the SINC field.
 - vi. Incrementing/decrementing or fixed address for the destination in the DINC field.
- (3) Write the starting source address in the SARx register for channel x.

Note: The values in the LLI.SARx register locations of each of the Linked List Items (LLIs) set up in memory, although fetched during an LLI fetch, are not used.
- (4) Write the channel configuration information into the CFGx register for channel x.
 - a) Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.

This step requires programming the HS_SEL_SRC/HS_SEL_DST bits. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.

- b) If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.
- (5) Make sure that the LLI.CTLx register locations of all LLI entries in memory (except the last) are set as shown in Row 7 of Table 9-19, while the LLI.CTLx register of the last Linked List Item must be set as described in Row 1 or Row 5 of Table 9-19. Fig 9-45 shows a Linked List example with two list items.
- (6) Ensure that the LLI.LLPx register locations of all LLI entries in memory (except the last) are non-zero and point to the next Linked List Item.
- (7) Ensure that the LLI.DARx register locations of all LLI entries in memory point to the start destination block address preceding that LLI fetch.
- (8) If DMAH_CHx_CTL_WB_EN = True, ensure that the LLI.CTLx.DONE field of the LLI.CTLx register locations of all LLIs in memory are cleared.
- (9) If source status fetching is enabled (DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and CFGx.SS_UPD_EN is enabled), program the SSTATARx register so that the source status information can be fetched from the location pointed to by the SSTATARx. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of Table 9-19.
- (10) If destination status fetching is enabled (DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and CFGx.DS_UPD_EN is enabled), program the DSTATARx register so that the destination status information can be fetched from the location pointed to by the DSTATARx register. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of Table 9-19.
- (11) If gather is enabled (DMAH_CHx_SRC_GAT_EN = True and CTLx.SRC_GATHER_EN is enabled), program the SGRx register for channel x.
- (12) If scatter is enabled (DMAH_CHx_DST_SCA_EN = True and CTLx.DST_SCATTER_EN is enabled) program the DSRx register for channel x.
- (13) Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, and ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
- (14) Program the CTLx and CFGx registers according to Row 7, as shown in Table 9-19.
- (15) Program the LLPx register with LLP(0), the pointer to the first linked list item.
- (16) Finally, enable the channel by writing a 1 to the ChEnReg.CH_EN bit; the transfer is performed. Ensure that bit 0 of the DmaCfgReg register is enabled.
- (17) The DMAC fetches the first LLI from the location pointed to by LLPx(0).
Note: The LLI.SARx, LLI.DARx, LLI.LLPx, and LLI.CTLx registers are fetched. The LLI.SARx register – although fetched – is not used.
- (18) Source and destination request single and burst DMAC transactions in order to transfer the block of data (assuming non-memory peripheral). The DMAC acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
- (19) Once the block of data is transferred, the source status information is fetched from the location pointed to by the SSTATARx register and stored in the SSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and CFGx.SS_UPD_EN is enabled. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of Table 9-19. The destination status information is fetched from the location pointed to by the DSTATARx register and stored in the DSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and CFGx.DS_UPD_EN is enabled. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of Table 9-19.
- (20) If DMAH_CHx_CTL_WB_EN = True, then the CTLx[63:32] register is written out to system memory. For conditions under which the CTLx[63:32] register is written out to system memory, refer to the Write Back column of Table 9-19.
The CTLx[63:32] register is written out to the same location on the same layer (LLPx.LMS) where it was originally fetched; that is, the location of the CTLx register of the linked list item fetched prior to the start of the block transfer. Only the second word of the CTLx register is written out – CTLx[63:32] – because only the CTLx.BLOCK_TS and CTLx.DONE fields have been updated by hardware within the DMAC. The CTLx.DONE bit is asserted to indicate block completion. Therefore, software can poll the LLI.CTLx.DONE bit of the CTLx register in the LLI to ascertain when a block transfer has completed.
Note: Do not poll the CTLx.DONE bit in the DMAC memory map. Instead, poll the LLI.CTLx.DONE bit in the LLI for that block. If the polled LLI.CTLx.DONE bit is asserted, then this block transfer has completed. This LLI.CTLx.DONE bit was cleared at the start of the transfer (Step 8).
- (21) The SSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and CFGx.SS_UPD_EN is enabled. It is written to the SSTATx register location of the LLI pointed to by the previously saved LLPx.LOC register. The DSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and CFGx.DS_UPD_EN is enabled. It is written to the DSTATx register location of the LLI pointed to by the previously saved LLPx.LOC register. The end-of-block interrupt, int_block, is generated after the write-back of the control and status registers has completed.
Note: The write-back location for the control and status registers is the LLI pointed to by the previous value of the LLPx.LOC register, not the LLI pointed to by the current value of the LLPx.LOC register.
- (22) The DMAC reloads the SARx register from the initial value. Hardware sets the block-complete interrupt. The DMAC samples the row number, as shown in Table 9-19. If the DMAC is in Row 1 or Row 5, then the DMA transfer has completed. Hardware sets the transfer

complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register ($\text{RawTfr}[n]$, n = channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, $\text{ClearTfr}[n]$, before the channel is enabled. If the DMAC is not in Row 1 or Row 5 as shown in Table 9-19, the following steps are performed.

(23) The DMA transfer proceeds as follows:

- a) If interrupts are enabled ($\text{CTLx.INT_EN} = 1$) and the block-complete interrupt is unmasked ($\text{MaskBlock}[x] = 1'b1$, where x is the channel number), hardware sets the block-complete interrupt when the block transfer has completed. It then stalls until the block-complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block-complete ISR (interrupt service routine) should clear the CFGx.RELOAD_SRC source reload bit. This puts the DMAC into Row 1, as shown in Table 9-19. If the next block is not the last block in the DMA transfer, then the source reload bit should remain enabled to keep the DMAC in Row 7, as shown in Table 9-19.
- b) If interrupts are disabled ($\text{CTLx.INT_EN} = 0$) or the block-complete interrupt is masked ($\text{MaskBlock}[x] = 1'b0$, where x is the channel number), then hardware does not stall until it detects a write to the block-complete interrupt clear register; instead, it immediately starts the next block transfer. In this case, software must clear the source reload bit, CFGx.RELOAD_SRC in order to put the device into Row 1 of Table 9-19 before the last block of the DMA transfer has completed.

(24) The DMAC fetches the next LLI from memory location pointed to by the current LLPx register and automatically reprograms the DARx , CTLx , and LLPx channel registers. Note that the SARx is not reprogrammed, since the reloaded value is used for the next DMA block transfer. If the next block is the last block of the DMA transfer, then the CTLx and LLPx registers just fetched from the LLI should match Row 1 or Row 5 of Table 9-19.

The DMA transfer might look like that shown in Fig 9-55.

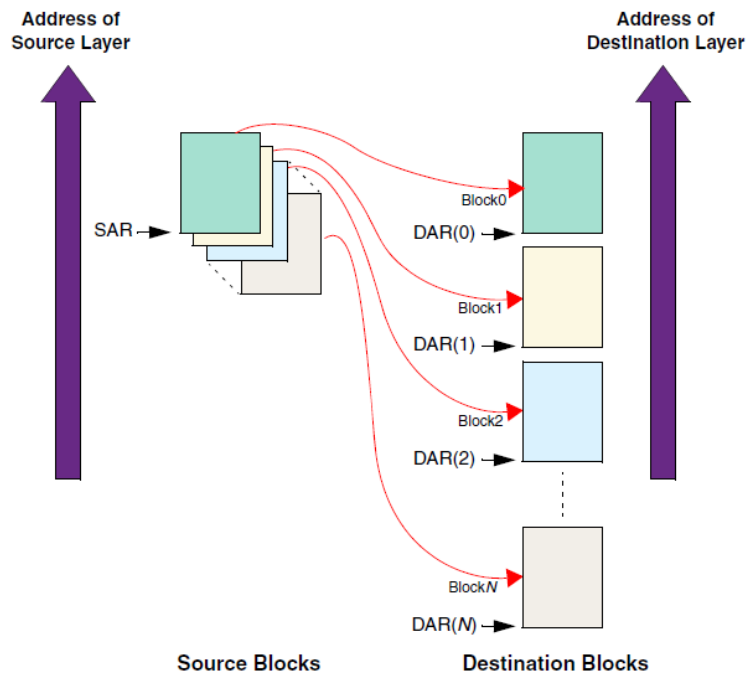


Fig 9-55 Multi-block DMA transfer with source address auto-reloaded and linked list destination address

The DMA transfer flow is shown in Fig 9-56.

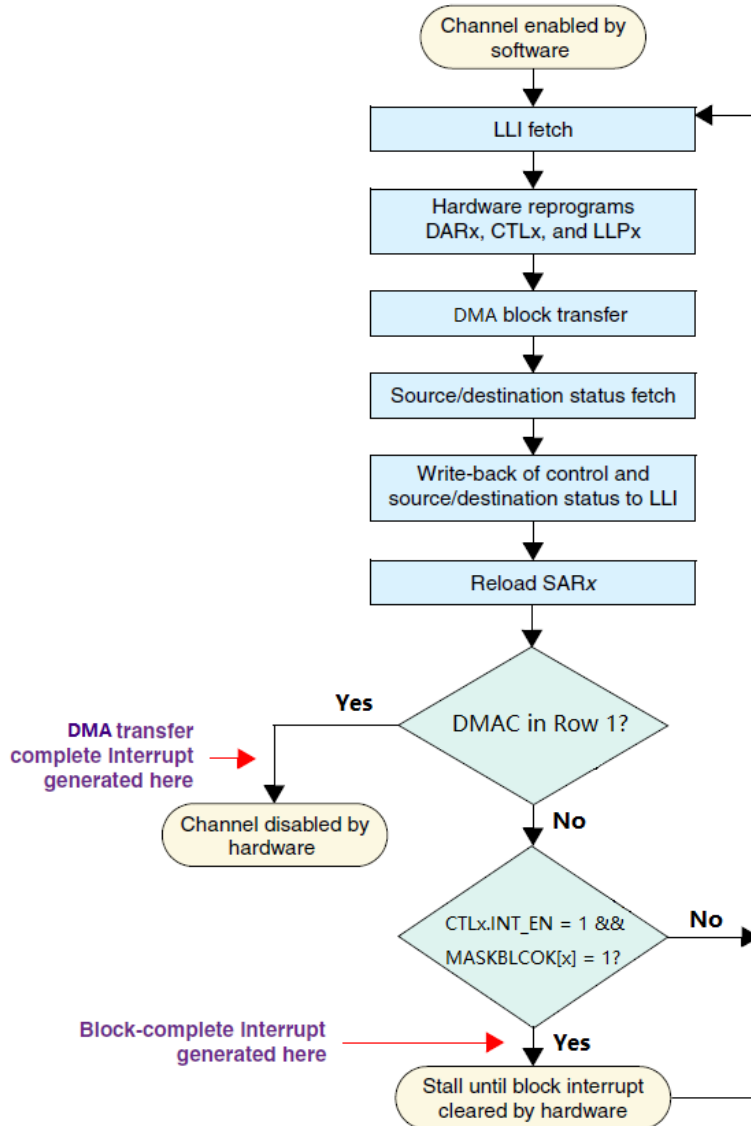


Fig 9-56 DMA transfer flow for source address auto-reloaded and linked list destination address

9.4.5.5 Multi-Block Transfer with Source Address Auto-Reloaded and Contiguous Destination Address (Row 3)

Note: This type of multi-block transfer can only be enabled when either of the following parameters is set:

- DMAH_CHx_MULTI_BLK_TYPE = 0
- DMAH_CHx_MULTI_BLK_TYPE = RELOAD_CONT

- (1) Read the Channel Enable register to choose a free (disabled) channel.
- (2) Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, and ClearErr. Reading the Interrupt Raw Status and Interrupt Status register confirms that all interrupts have been cleared.
- (3) Program the following channel registers:
 - a) Write the starting source address in the SARx register for channel x.
 - b) Write the starting destination address in the DARx register for channel x.
 - c) Program CTLx and CFGx according to Row 3, as shown in Table 9-19. Program the LLPx register with 0.
 - d) Write the control information for the DMA transfer in the CTLx register for channel x. For example, in the register, you can program the following:

- i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register. Table 9-17 lists the decoding for this field.
- ii. Set up the transfer characteristics, such as:
 - Transfer width for the source in the SRC_TR_WIDTH field; Table 9-16 lists the decoding for this field.
 - Transfer width for the destination in the DST_TR_WIDTH field; Table 9-16 lists the decoding for this field.
 - Source master layer in the SMS field where the source resides.
 - Destination master layer in the DMS field where the destination resides.
 - Incrementing/decrementing or fixed address for the source in the SINC field.
 - Incrementing/decrementing or fixed address for the destination in the DINC field.
- e) If gather is enabled (DMAH_CHx_SRC_GAT_EN = True and CTLx.SRC_GATHER_EN is enabled), program the SGRx register for channel x.
- f) If scatter is enabled (DMAH_CHx_DST_SCA_EN = True and CTLx.DST_SCATTER_EN), program the DSRx register for channel x.
- g) Write the channel configuration information into the CFGx register for channel x.
 - i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.
 - ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.
- (4) After the DMAC channel has been programmed, enable the channel by writing a 1 to the ChEnReg.CH_EN bit. Ensure that bit 0 of the DmaCfgReg register is enabled.
- (5) Source and destination request single and burst DMAC transactions to transfer the block of data (assuming non-memory peripherals). The DMAC acknowledges on completion of each burst/single transaction and carries out the block transfer.
- (6) When the block transfer has completed, the DMAC reloads the SARx register; the DARx register remains unchanged. Hardware sets the block-complete interrupt. The DMAC then samples the row number, as shown in Table 9-19. If the DMAC is in Row 1, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (RawTfr[n], n = channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, ClearTfr[n], before the channel is enabled. If the DMAC is not in Row 1, the next step is performed.
- (7) The DMA transfer proceeds as follows:
 - a) If interrupts are enabled (CTLxx.INT_EN = 1) and the block-complete interrupt is unmasked (MaskBlock[x] = 1'b1, where x is the channel number), hardware sets the block-complete interrupt when the block transfer has completed. It then stalls until the block-complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block-complete ISR (interrupt service routine) should clear the source reload bit, CFGx.RELOAD_SRC. This puts the DMAC into Row 1, as shown in Table 9-19. If the next block is not the last block in the DMA transfer, then the source reload bit should remain enabled to keep the DMAC in Row 3, as shown in Table 9-19.
 - b) If interrupts are disabled (CTLxx.INT_EN = 0) or the block-complete interrupt is masked (MaskBlock[x] = 1'b0, where x is the channel number), then hardware does not stall until it detects a write to the block-complete interrupt clear register; instead, it starts the next block transfer immediately. In this case, software must clear the source reload bit, CFGx.RELOAD_SRC, to put the device into Row 1 of Table 9-19 before the last block of the DMA transfer has completed.

The transfer is similar to that shown in Fig 9-57.

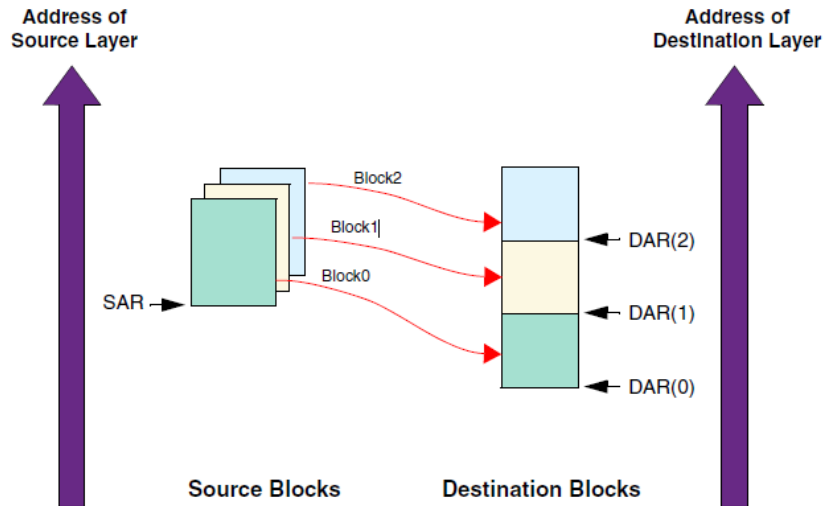


Fig 9-57 Multi-block DMA transfer with source address auto-reloaded and contiguous destination address

The DMA transfer flow is shown in Fig 9-58.

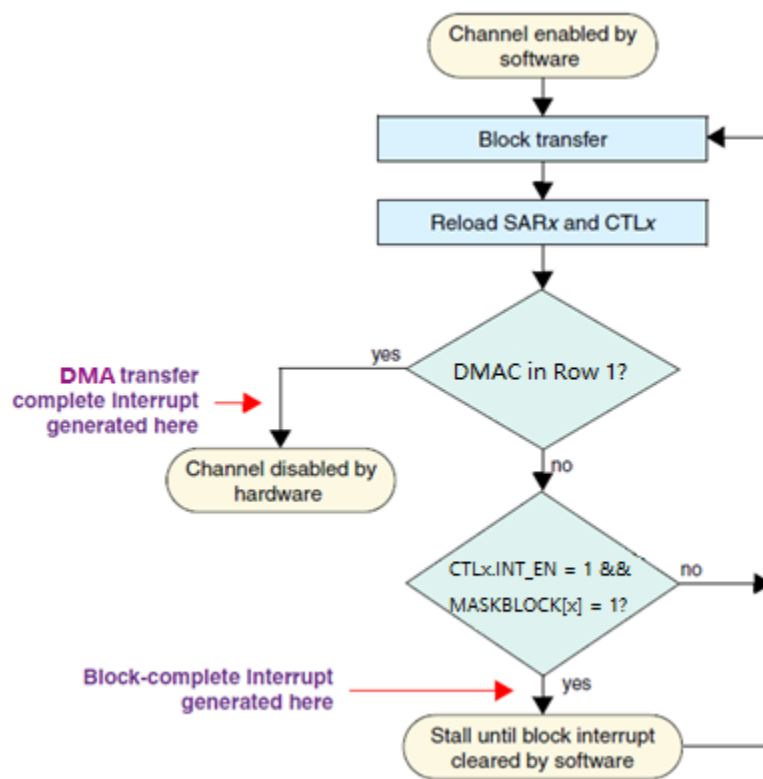


Fig 9-58 DMA transfer flow for source address auto-reloaded and contiguous destination address

9.4.5.6 Multi-Block DMA Transfer with Linked List for Source and Contiguous Destination Address (Row 8)

Note: This type of multi-block transfer can only be enabled when either of the following parameters is set:

- DMAH_CHx_MULTI_BLK_TYPE = 0
- DMAH_CHx_MULTI_BLK_TYPE = LLP_CONT

- (1) Read the Channel Enable register in order to choose a free (disabled) channel.
- (2) Set up the linked list in memory. Write the control information in the LLI.CTLx register location of the block descriptor for each LLI in memory (see Fig 9-45) for channel x. For example, in the register, you can program the following:
 - a) Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the CTLx register. Table 9-17 lists the decoding for this field.
 - b) Set up the transfer characteristics, such as:
 - i. Transfer width for the source in the SRC_TR_WIDTH field. Table 9-16 lists the decoding for this field.
 - ii. Transfer width for the destination in the DST_TR_WIDTH field. Table 9-16 lists the decoding for this field.
 - iii. Source master layer in the SMS field where the source resides.
 - iv. Destination master layer in the DMS field where the destination resides.
 - v. Incrementing/decrementing or fixed address for the source in the SINC field.
 - vi. Incrementing/decrementing or fixed address for the destination in the DINC field.
- (3) Write the starting source address in the DARx register for channel x.
Note: The values in the LLI.DARx register locations of each of the Linked List Items (LLIs) set up in memory, although fetched during an LLI fetch, are not used.
- (4) Write the channel configuration information into the CFGx register for channel x.
 - a) Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
 This step requires programming the HS_SEL_SRC/HS_SEL_DST bits. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.
 - b) If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DEST_PER bits, respectively.
- (5) Ensure that the LLI.CTLx register locations of all LLI (except the last) are set as shown in Row 8 of Table 9-19, while the LLI.CTLx register of the last Linked List Item must be set as described in Row 1 or Row 5 of Table 9-19. Fig 9-45 shows a Linked List example with two list items.
- (6) Ensure that the LLI.LLPx register locations of all LLI entries in memory (except the last) are non-zero and point to the next Linked List Item.
- (7) Ensure that the LLI.SARx register locations of all LLI entries in memory point to the start destination block address preceding that LLI fetch.
- (8) If DMAH_CHx_CTL_WB_EN = True, ensure that the LLI.CTLx.DONE field of the LLI.CTLx register locations of all LLIs in memory are cleared.
- (9) If source status fetching is enabled (DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and CFGx.SS_UPD_EN is enabled), program the SSTATARx register so that the source status information can be fetched from the location pointed to by the SSTATARx. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of Table 9-19.
- (10) If destination status fetching is enabled (DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and CFGx.DS_UPD_EN is enabled), program the DSTATARx register so that the destination status information can be fetched from the location pointed to by the DSTATARx register. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of Table 9-19.
- (11) If gather is enabled (DMAH_CHx_SRC_GAT_EN = True and CTLx.SRC_GATHER_EN is enabled), program the SGRx register for channel x.
- (12) If scatter is enabled (DMAH_CHx_DST_SCA_EN = True and CTLx.DST_SCATTER_EN is enabled) program the DSRx register for channel x.
- (13) Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, and ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
- (14) Program the CTLx and CFGx registers according to Row 8, as shown in Table 9-19.
- (15) Program the LLPx register with LLPx(0), the pointer to the first linked list item.
- (16) Finally, enable the channel by writing a 1 to the ChEnReg.CH_EN bit; the transfer is performed. Ensure that bit 0 of the DmaCfgReg register is enabled.
- (17) The DMAC fetches the first LLI from the location pointed to by LLPx(0).
Note: The LLI.SARx, LLI.DARx, LLI.LLPx, and LLI.CTLx registers are fetched. The LLI.DARx register – although fetched – is not used. The DARx register in the DMAC remains unchanged.
- (18) Source and destination request single and burst DMAC transactions in order to transfer the block of data (assuming non-memory peripheral). The DMAC acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
- (19) Once the block of data is transferred, the source status information is fetched from the location pointed to by the SSTATARx register and stored in the SSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and CFGx.SS_UPD_EN is enabled. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of Table 9-19. The destination status information is fetched from the location pointed to by the DSTATARx register and stored in the DSTATx register if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and CFGx.DS_UPD_EN is enabled. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of Table 9-19.

- (20) If DMAH_CHx_CTL_WB_EN = True, then the CTLx[63:32] register is written out to system memory. For conditions under which the CTLx[63:32] register is written out to system memory, refer to the Write Back column of Table 9-19.
- The CTLx[63:32] register is written out to the same location on the same layer (LLPx.LMS) where it was originally fetched; that is, the location of the CTLx register of the linked list item fetched prior to the start of the block transfer. Only the second word of the CTLx register is written out, CTLx[63:32], because only the CTLx.BLOCK_TS and CTLx.DONE fields have been updated by hardware within the DMAC. Additionally, the CTLx.DONE bit is asserted to indicate block completion. Therefore, software can poll the LLI.CTLx.DONE bit field of the CTLx register in the LLI to ascertain when a block transfer has completed.
- Note:** Do not poll the CTLx.DONE bit in the DMAC memory map. Instead, poll the LLI.CTLx.DONE bit in the LLI for that block. If the polled LLI.CTLx.DONE bit is asserted, then this block transfer has completed. This LLI.CTLx.DONE bit was cleared at the start of the transfer (Step 8).
- (21) The SSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_SRC = True, and CFGx.SS_UPD_EN is enabled. It is written to the SSTATx register location of the LLI pointed to by the previously saved LLPx.LOC register. The DSTATx register is now written out to system memory if DMAH_CHx_CTL_WB_EN = True, DMAH_CHx_STAT_DST = True, and CFGx.DS_UPD_EN is enabled. It is written to the DSTATx register location of the LLI pointed to by the previously saved LLPx.LOC register. The end-of-block interrupt, int_block, is generated after the write-back of the control and status registers has completed.
- Note:** The write-back location for the control and status registers is the LLI pointed to by the previous value of the LLPx.LOC register, not the LLI pointed to by the current value of the LLPx.LOC register.
- (22) The DMAC does not wait for the block interrupt to be cleared, but continues and fetches the next LLI from the memory location pointed to by the current LLPx register and automatically reprograms the SARx, CTLx, and LLPx channel registers. The DARx register is left unchanged. The DMA transfer continues until the DMAC samples that the CTLx and LLPx registers at the end of a block transfer match those described in Row 1 or Row 5 of Table 9-15 (as discussed earlier). The DMAC then knows that the previously transferred block was the last block in the DMA transfer.

The DMA transfer might look like that shown in Fig 9-59. Note that the destination address is decrementing.

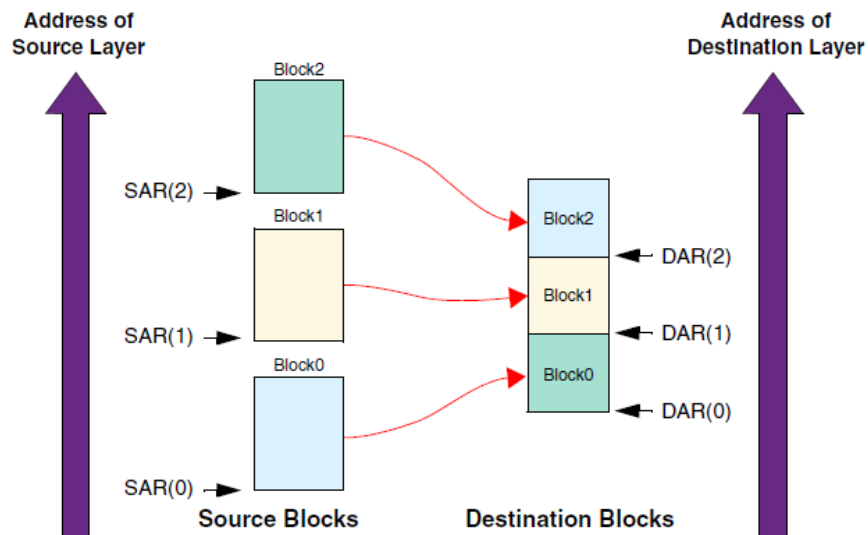


Fig 9-59 Multi-block DMA transfer with linked list source address and contiguous destination address

The DMA transfer flow is shown in Fig 9-60.

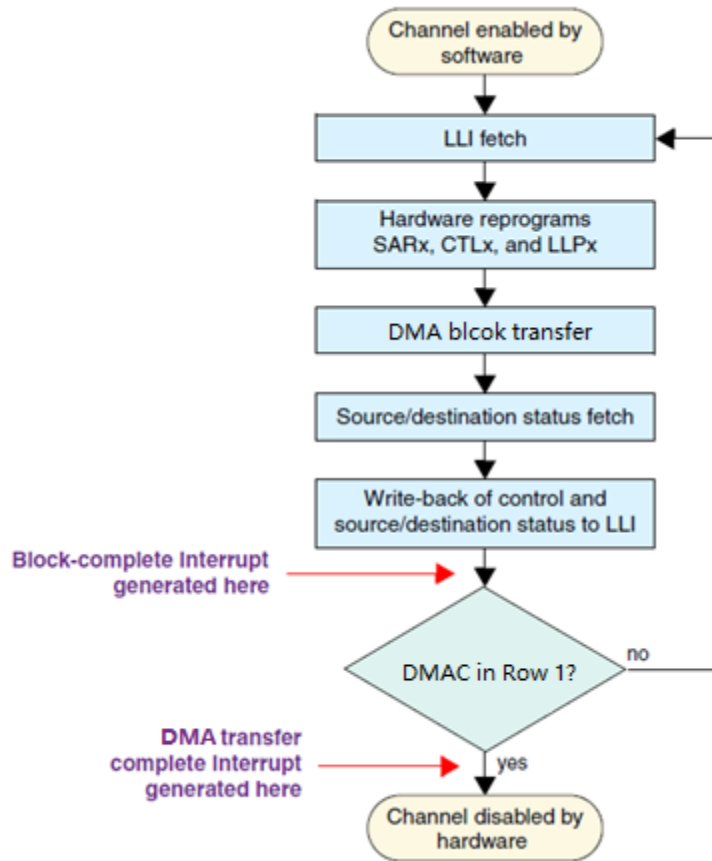


Fig 9-60 DMA transfer flow for source address auto-reloaded and contiguous destination address

9.4.6 Disabling a Channel Prior to Transfer Completion

Under normal operation, software enables a channel by writing a 1 to the channel enable register, ChEnReg.CH_EN, and hardware disables a channel on transfer completion by clearing the ChEnReg.CH_EN register bit.

The recommended way for software to disable a channel without losing data is to use the CH_SUSP bit in conjunction with the FIFO_EMPTY bit in the Channel Configuration Register (CFGx).

- (1) If software wishes to disable a channel prior to the DMA transfer completion, then it can set the CFGx.CH_SUSP bit to tell the DMAC to halt all transfers from the source peripheral. Therefore, the channel FIFO receives no new data.
- (2) Software can now poll the CFGx.FIFO_EMPTY bit until it indicates that the channel FIFO is empty.
- (3) The ChEnReg.CH_EN bit can then be cleared by software once the channel FIFO is empty.

When CTLx.SRC_TR_WIDTH < CTLx.DST_TR_WIDTH and the CFGx.CH_SUSP bit is high, the CFGx.FIFO_EMPTY is asserted once the contents of the FIFO do not permit a single word of CTLx.DST_TR_WIDTH to be formed. However, there may still be data in the channel FIFO, but not enough to form a single transfer of CTLx.DST_TR_WIDTH. In this scenario, once the channel is disabled, the remaining data in the channel FIFO is not transferred to the destination peripheral.

It is permissible to remove the channel from the suspension state by writing a 0 to the CFGx.CH_SUSP register. The DMA transfer completes in the normal manner.

Note:

- If a channel is disabled by software, an active single or burst transaction is not guaranteed to receive an acknowledgement.
- If the DMAC is configured to use defined length bursts (DMAH_INCR_BURSTS = 0), disabling the channel via software prior to completing a transfer is not supported.

9.4.6.1 Abnormal Transfer Termination

A DMAC DMA transfer may be terminated abruptly by software by clearing the channel enable bit, ChEnReg.CH_EN. You must not assume that the channel is disabled immediately after the ChEnReg. The CH_EN bit is cleared over the AHB slave interface. Consider this as a request to disable the channel. You must poll ChEnReg.CH_EN and confirm that the channel is disabled by reading back 0. A case where the channel is not disabled after a channel disable request is where either the source or destination has received a split or retry response. The DMAC must keep re-attempting the transfer to the system HADDR that originally received the split or retry response until an OKAY response is returned; to do otherwise is an AMBA protocol violation.

Software may terminate all channels abruptly by clearing the global enable bit in the DMAC Configuration Register (DmaCfgReg[0]). Again, you must not assume that all channels are disabled immediately after the DmaCfgReg[0] is cleared over the AHB slave interface. Consider this as a request to disable all channels. You must poll ChEnReg and confirm that all channels are disabled by reading back 0.

Note:

- If the channel enable bit is cleared while there is data in the channel FIFO, this data is not sent to the destination peripheral and is not present when the channel is re-enabled. For read-sensitive source peripherals, such as a source FIFO, this data is therefore lost. When the source is not a read-sensitive device (such as memory), disabling a channel without waiting for the channel FIFO to empty may be acceptable, since the data is available from the source peripheral upon request and is not lost.
- If a channel is disabled by software, an active single or burst transaction is not guaranteed to receive an acknowledgement.
- If the DMAC is configured to use defined length bursts (DMAH_INCR_BURSTS = 0), disabling the channel via software prior to completing a transfer is not supported.

9.4.7 Defined-Length Burst Support on DMAC

By default, the DMAC support incremental (INCR) bursts only. To achieve better performance, defined length bursts, such as INCR4, INCR8 and INCR16 are required. The DMAC can be configured to use defined-length bursts by setting the configuration parameter DMAH_INCR_BURSTS to 0. In this mode, the DMAC will select the largest valid defined-length burst to complete the transfer.

10 General Timers

10.1 Basic Timer

10.1.1 Introduction

The basic timers (TIM0/TIM1/TIM2/TIM3) consist of a 32-bit auto-reload counter without prescaler. They may be used as generic timers for time-base generation.

10.1.2 Features

The features of basic timers are listed in Table 10-1.

Table 10-1 Basic timers features

Name	KM4: TIM0/TIM1/TIM2/TIM3	KM0: TIM00/TIM01/TIM02/TIM03
Channels	1	1
Clock source	32kHz	32kHz
Resolution	32-bit	32-bit
Prescaler	-	-
Counter mode	Up	Up
One pulse mode	-	-
PWM mode with polarity selection	-	-
Statistic pulse width	-	-
Statistic pulse number	-	-
Interrupt generation	●	●
Input pin	-	-
Output pin	-	-

10.1.3 Block Diagram

The block diagram of the basic timers is shown in Fig 10-1.

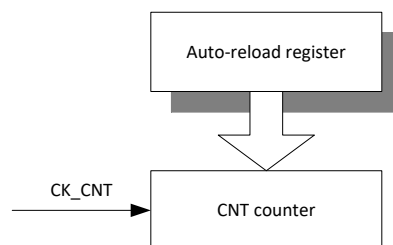


Fig 10-1 Block diagram

10.1.4 Functional Description

10.1.4.1 Upcounting Mode

This timer is a 32-bit counter with its related auto-reload register. The counter can count up.

The counter, also the auto-reload register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The contents of the preload register are transferred into the shadow register permanently or at each update event (UEV) depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR register. The update event is sent when the counter reaches the overflow; or if the UDIS bit equals to 0 in the TIMx_CR register, it can also be generated by software.

In upcounting mode, the counter counts from 0 to the auto-reload value (the content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register by software also generates an update event.

The update event can be disabled by software by setting the UDIS bit in the TIMx_CR register. This is to avoid updating the shadow registers while writing new values to the preload register. No update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0. In addition, if the URS bit in the TIMx_CR register is set, setting the UG bit generates an update event but without setting the update interrupt flag (UIF), thus no interrupt is sent.

When an update event occurs, all the registers are updated and the update flag (the UIF bit in TIMx_SR) is set depending on the URS bit. The auto-reload shadow register is updated with the preload value (TIMx_ARR).

10.1.4.2 Interrupt Event

The interrupt event is configured by TIMx_DIER.

When the UIF bit in the TIMx_SR register is asserted, the according interrupt event is asserted, which is determined by the configuration of TIMx_DIER. And when the counter counts overflow, the UIF is set automatically.

10.2 Pulse Mode Timer

10.2.1 Introduction

The pulse timer (TIM4) consists of a 16-bit auto-reload counter driven by a 8-bit programmable prescaler. It can be used for a variety of purposes, including measuring the pulse lengths or numbers of input signals.

10.2.2 Features

The features of the pulse timer are listed in Table 10-2.

Table 10-2 Pulse timer features

Name	KM4: TIM4	KM0: TIM04
Channels	1	1
Clock source	XTAL	XTAL
Resolution	16-bit	16-bit
Prescaler	8-bit	8-bit
Counter mode	Up	Up
One pulse mode	-	-
PWM mode with polarity selection	-	-
Statistic pulse width	●	●
Statistic pulse number	●	●
Interrupt generation	●	●

Input pin	2 input capture	2 input capture
Output pin	-	-

10.2.3 Block Diagram

The block diagram of the pulse timer (TIM4) is shown in Fig 10-2.

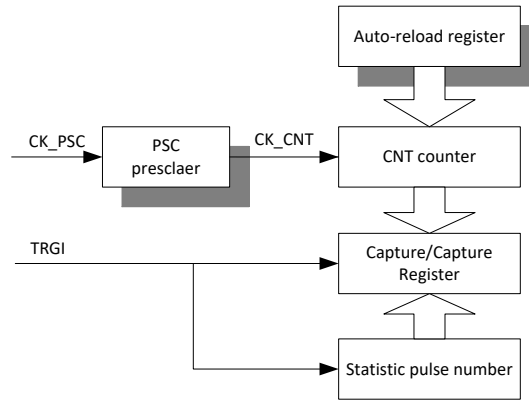


Fig 10-2 TIM4 block diagram

10.2.4 Functional Description

10.2.4.1 Upcounting Mode

This timer is a 16-bit counter with its related auto-reload register. The counter can count up. The counter clock can be divided by a 8-bit prescaler.

The counter, also the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The contents of the preload register are transferred into the shadow register permanently or at each update event depending on the ARPE bit in the TIMx_CR register. The update event is sent when the counter reaches the overflow; or if the UDIS bit equals to 0 in the TIMx_CR register, it can also be generated by software.

The prescaler can divide the counter clock frequency by any factor between 1 and 256. It is based on a 8-bit counter controlled through a 8-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

In upcounting mode, the counter counts from 0 to the auto-reload value (the content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register by software also generates an update event.

The update event can be disabled by software by setting the UDIS bit in the TIMx_CR register. This is to avoid updating the shadow registers while writing new values to the preload registers. No update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler, but the prescale rate doesn't change. In addition, if the URS bit in the TIMx_CR register is set,

setting the UG bit generates an update event but without setting the UIF flag, thus no interrupt is sent. This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (the UIF bit in TIMx_SR) is set depending on the URS bit:

- The auto-reload shadow register is updated with the preload value (TIMx_ARR).
- The buffer of the prescaler is reloaded with the preload value (the content of the TIMx_PSC register).

10.2.4.2 Statistic Pulse Width

In pulse mode 0, setting '0' in the CCOM bit of the TIMx_CCR0 register, the pulse timer can count the width of active level of TRGI. When the TRGI is transferred to active level from inactive level, the counter is enabled automatically. When the TRGI is transferred to inactive level from active level, the counter is disabled automatically, the CCOIF is set and the current counter value is copied to CCR0 field of the TIMx_CCR0 register.

Fig 10-3 gives an example of statistic pulse width mode when prescaler division is 1 and positive edge of TRGI is active for capture.

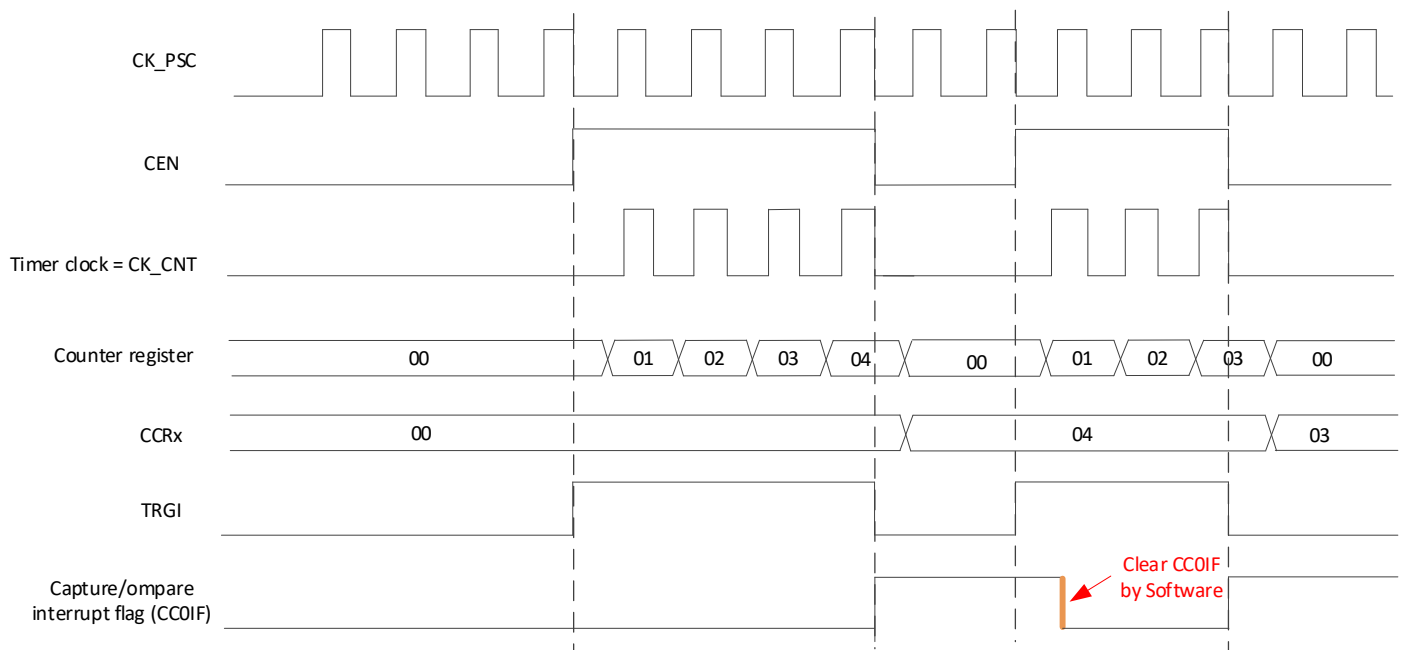
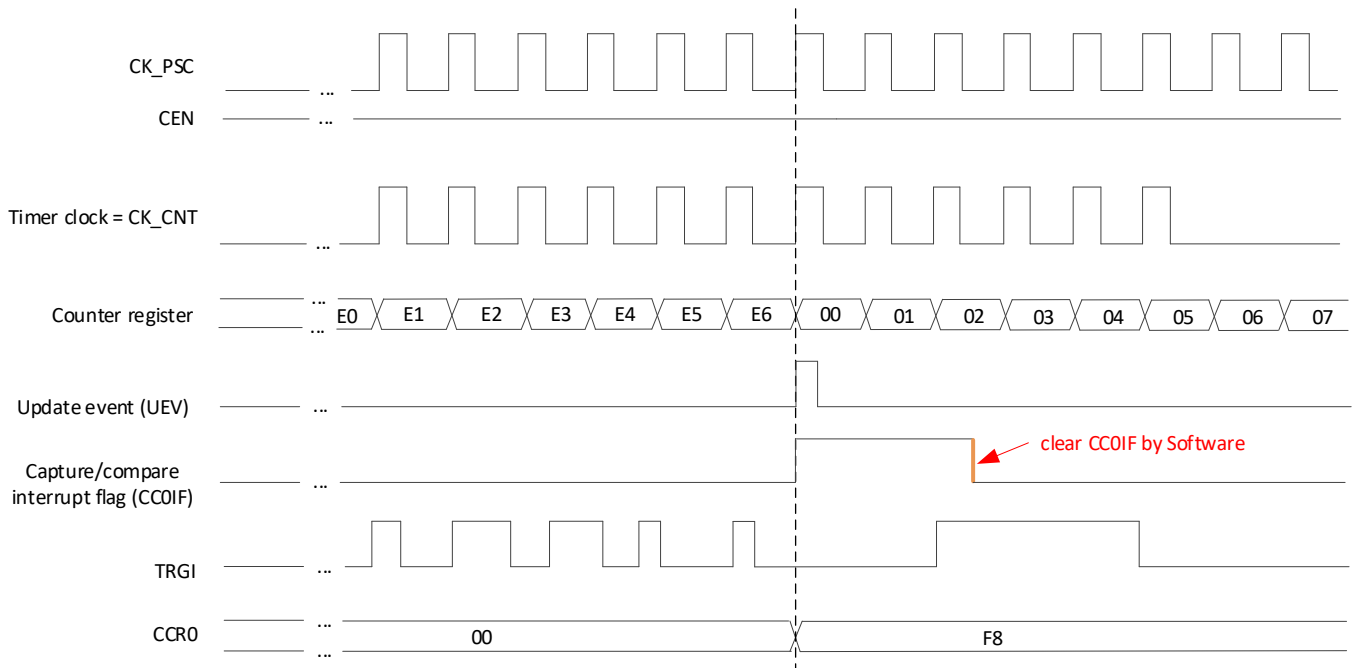


Fig 10-3 Statistic pulse width mode diagram (positive edge of TRGI is active for capture)

10.2.4.3 Statistic Pulse Number

In pulse mode 1, setting '1' in the CCOM bit of the TIMx_CCR0 register, the pulse timer can count the number of active edge of TRGI in the given period. When the counter overflows, the CCOIF is set and the number is copied to CCR0 field of the TIMx_CCR0 register.

Fig 10-4 gives an example of statistic pulse number mode when prescaler division is 1, positive edge of TRGI is active for capture, and the ARR field equals to E6.



Note: The register TIMx_CCR0 captures the pulse number F8 when the counter is overflowed (UEV).

Fig 10-4 Statistic pulse number mode diagram (positive edge of TRGI is active for capture, ARR=E6)

10.3 PWM Mode Timer

10.3.1 Introduction

The PWM timer (TIM5) consists of a 16-bit auto-reload counter driven by a 8-bit programmable prescaler. It can be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler.

10.3.2 Features

The features of the PWM timer (TIM5) are listed in Table 10-3.

Table 10-3 PWM timer features

Name	KM4: TIM5	KM0: TIM05
Channels	18	6
Clock source	XTAL	XTAL
Resolution	16-bit	16-bit
Prescaler	8-bit	8-bit
Counter mode	Up	Up
One-pulse mode	●	●
PWM mode with polarity selection	●	●
Statistic pulse width	-	-
Statistic pulse number	-	-
Interrupt generation	●	●
Input pin	2 input capture	2 input capture
Output pin	18 PWM out	6 PWM out
Sleep mode	-	●

0%/100%

10.3.3 Block Diagram

The block diagram of the PWM timer (TIM5) is shown in Fig 10-5.

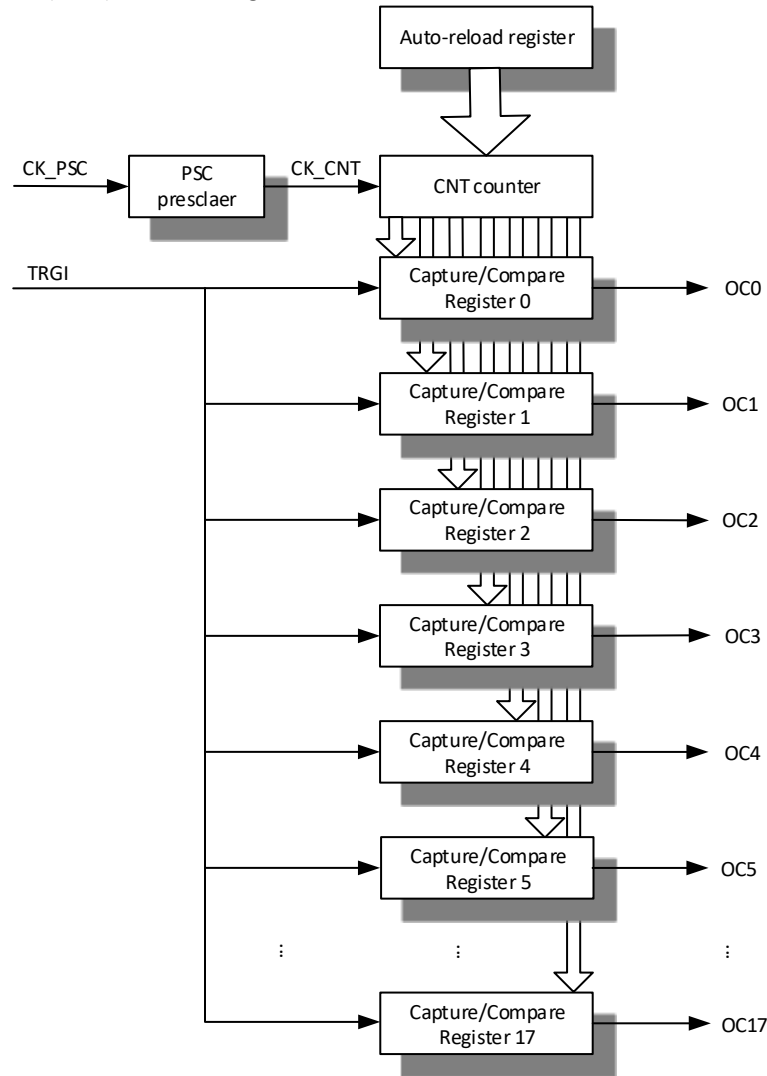


Fig 10-5 PWM timer block diagram

10.3.4 Functional Description

10.3.4.1 Upcounting Mode

This timer is a 16-bit counter with its related auto-reload register. The counter can count up. The counter clock can be divided by a 8-bit prescaler.

The counter, also the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)

- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The contents of the preload register are transferred into the shadow register permanently or at each update event depending on the ARPE bit in the TIMx_CR register. The update event is sent when the counter reaches the overflow; or if the UDIS bit equals to 0 in the TIMx_CR register, it can also be generated by software.

The prescaler can divide the counter clock frequency by any factor between 1 and 256. It is based on a 8-bit counter controlled through a 8-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Fig 10-6 and Fig 10-7 give some examples of the counter behavior when the prescaler ratio is changed on the fly.

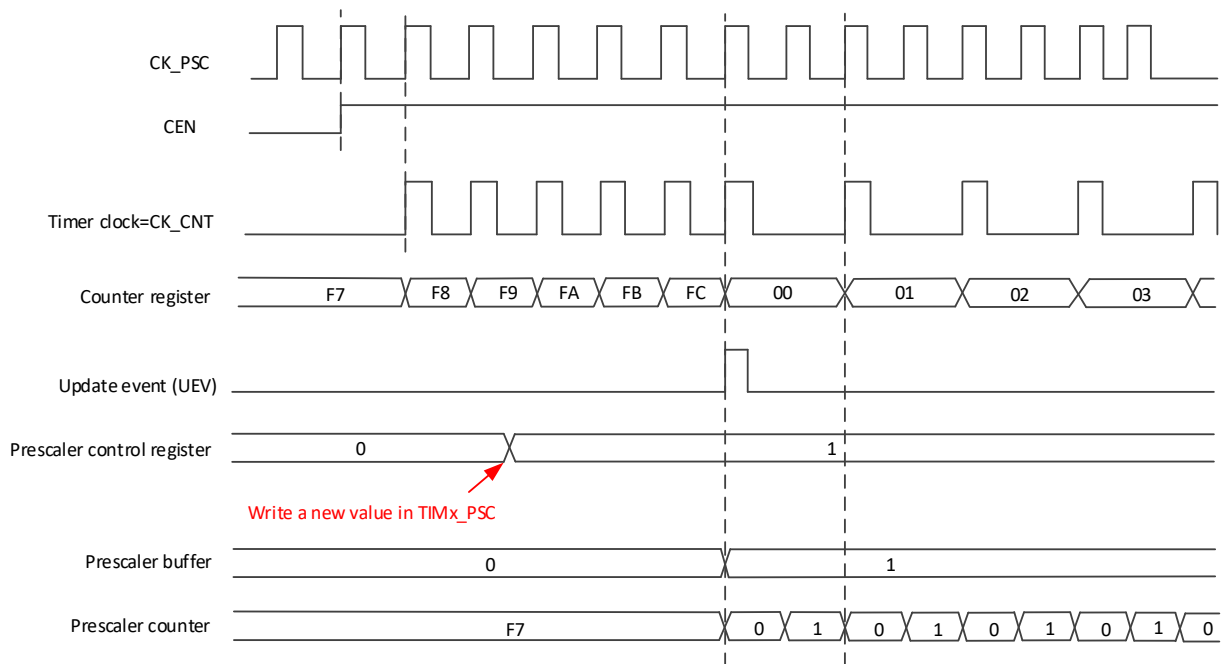


Fig 10-6 Counter timing diagram with prescaler division change from 1 to 2

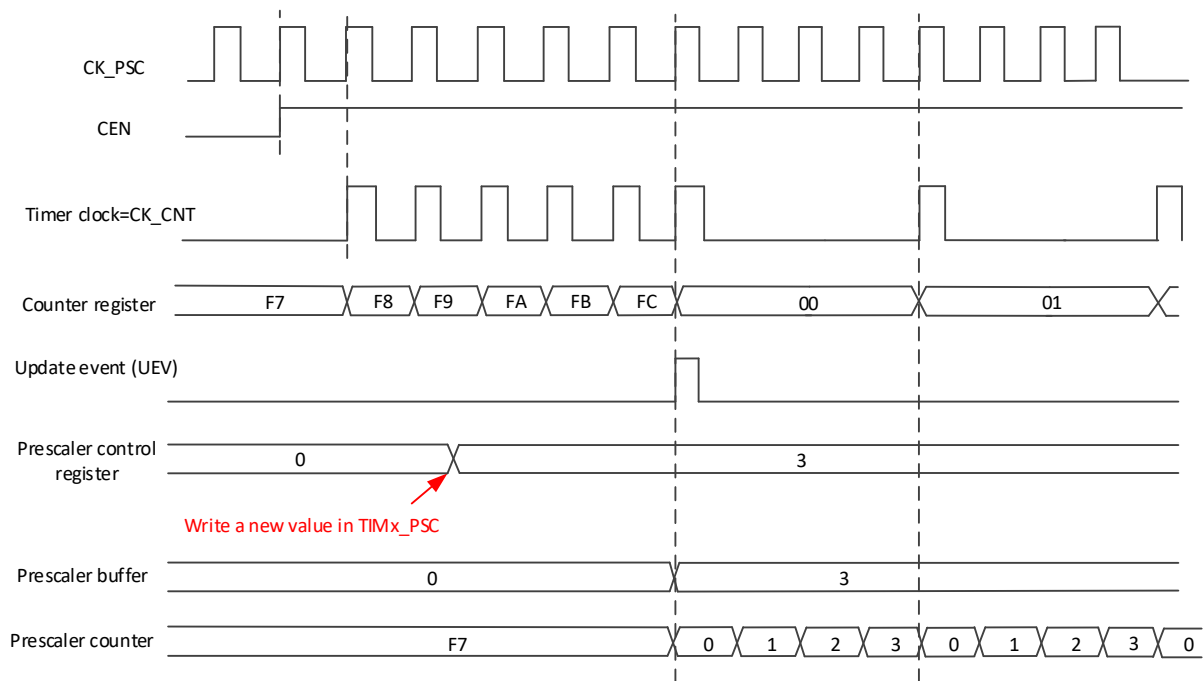


Fig 10-7 Counter timing diagram with prescaler division change from 1 to 4

In upcounting mode, the counter counts from 0 to the auto-reload value (the content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register by software also generates an update event.

The update event can be disabled by software by setting the UDIS bit in the TIMx_CR register. This is to avoid updating the shadow registers while writing new values to the preload registers. No update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler, but the prescale rate doesn't change. In addition, if the URS bit in the TIMx_CR register is set, setting the UG bit generates an update event but without setting the UIF flag, thus no interrupt is sent. This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (the UIF bit in TIMx_SR) is set depending on the URS bit:

- The auto-reload shadow register is updated with the preload value (TIMx_ARR).
- The buffer of the prescaler is reloaded with the preload value (the content of the TIMx_PSC register)

Fig 10-8 to Fig 10-13 show some examples of the counter behavior for different clock frequencies when the ARR field equals to 0x36.

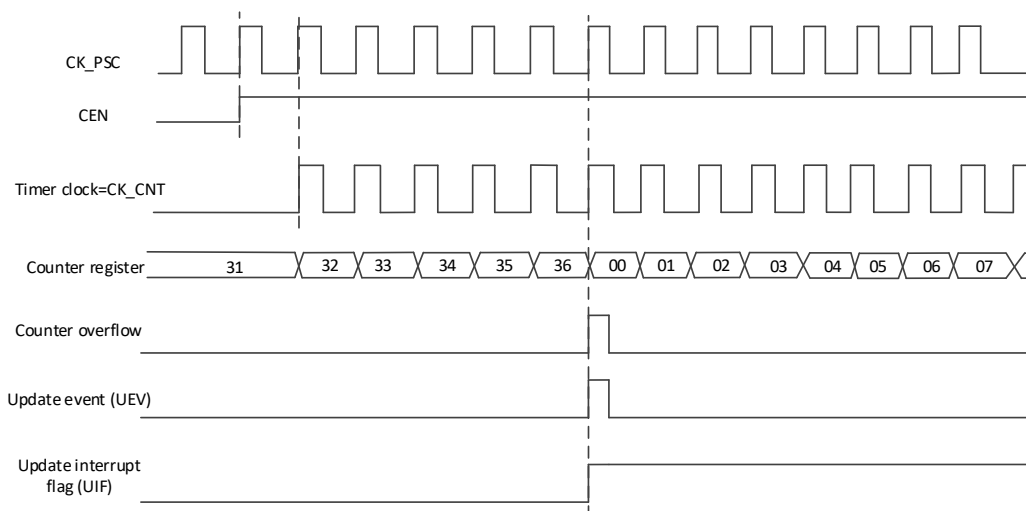


Fig 10-8 Counter timing diagram (internal clock divided by 1)

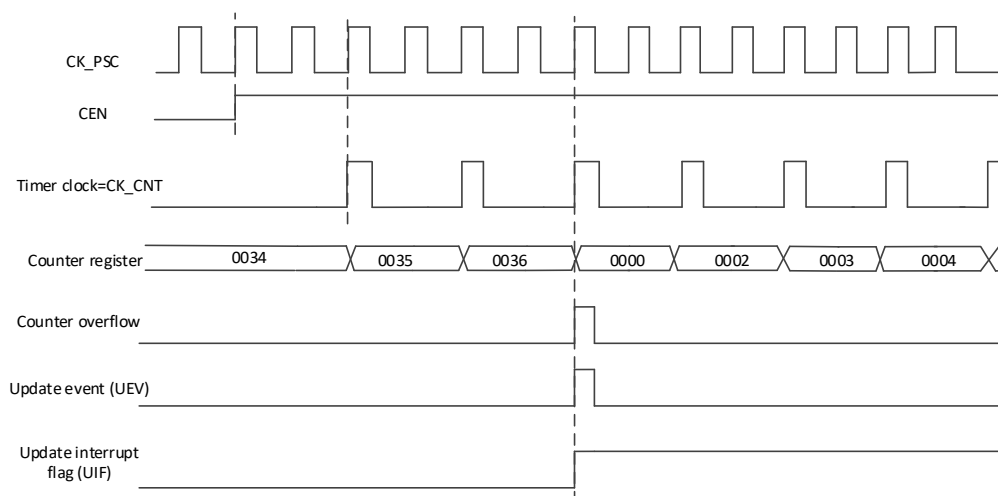


Fig 10-9 Counter timing diagram (internal clock divided by 2)

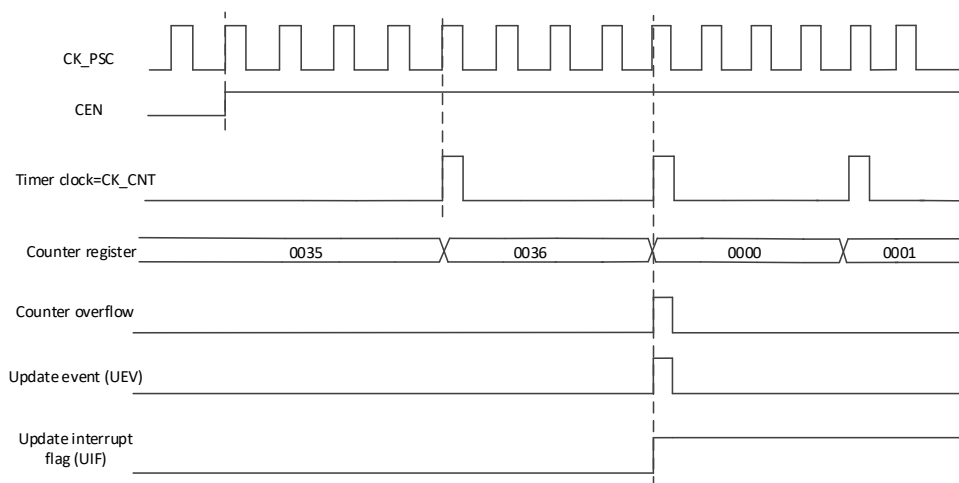


Fig 10-10 Counter timing diagram (internal clock divided by 4)

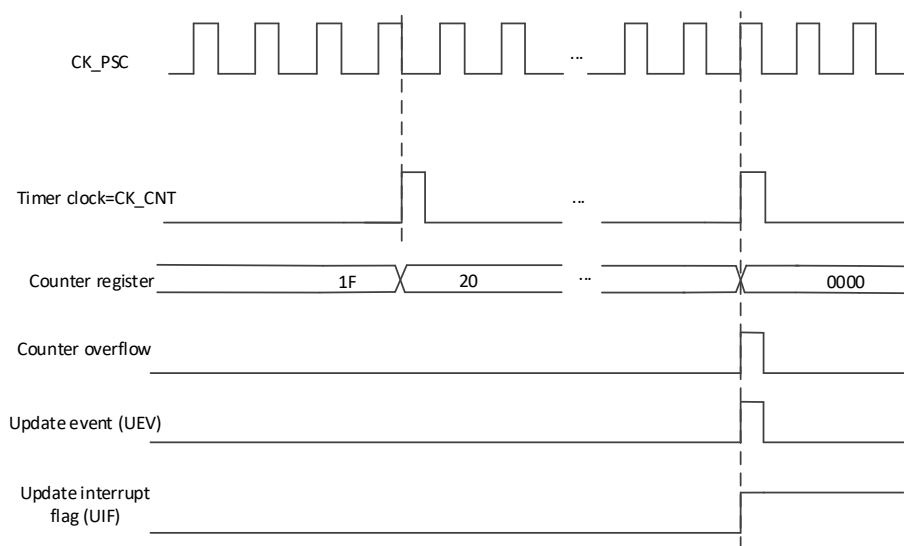


Fig 10-11 Counter timing diagram (internal clock divided by N)

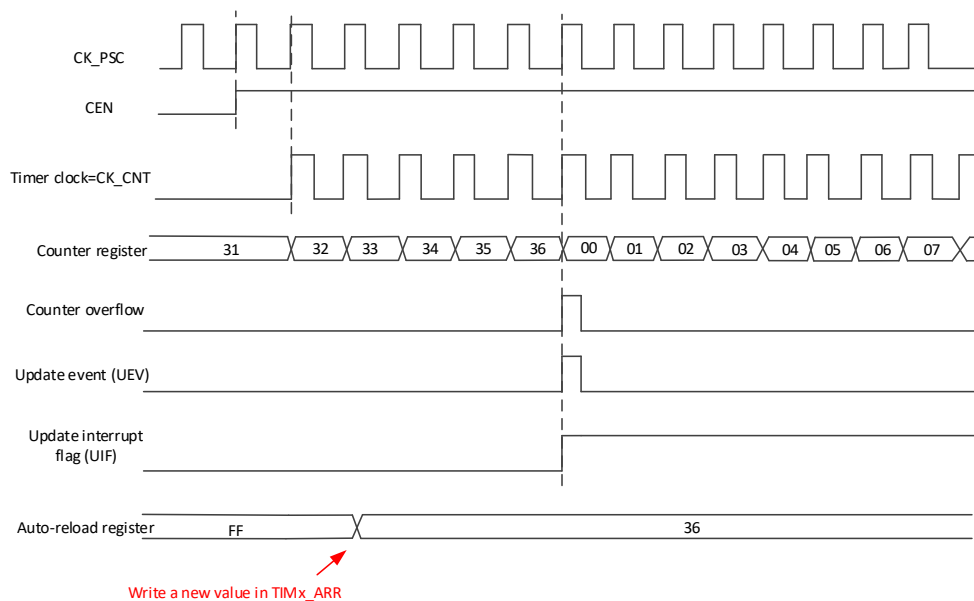


Fig 10-12 Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

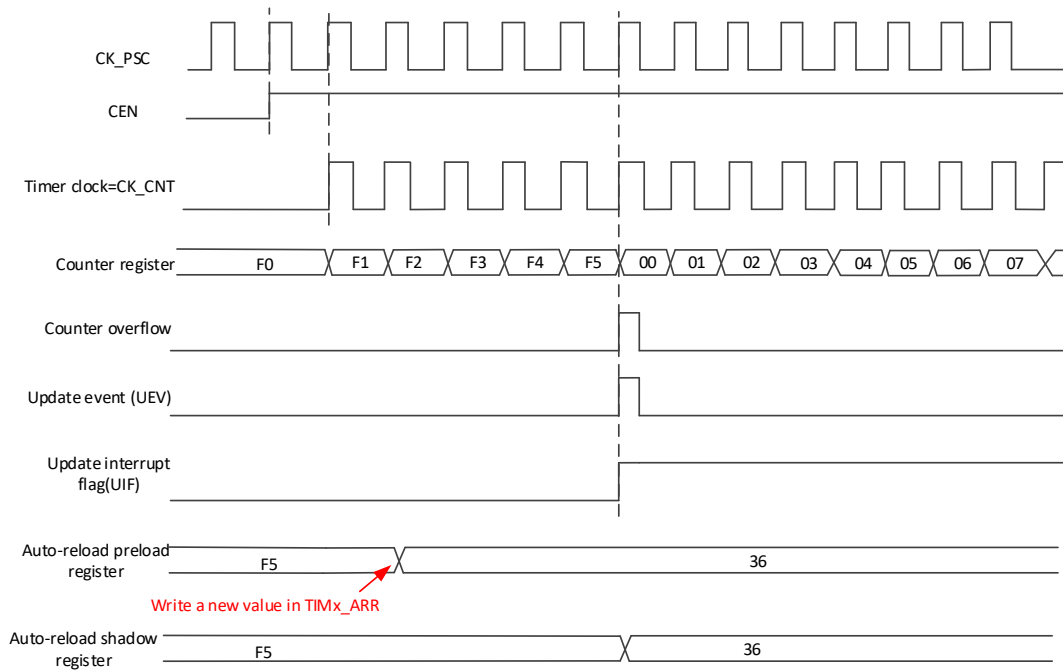


Fig 10-13 Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)

10.3.4.2 Input Capture Mode

In input capture mode, setting '1' in the CCxM bit of the TIMx_CCRx register, the CCRx field of capture/compare Registers (TIMx_CCRx) is used to latch the value of the counter after a transition detected by the TRGI signal. When a capture occurs, the corresponding CCxIF flag in the TIMx_SR register is set and an interrupt can be sent if they are enabled. CCxIF can be cleared by software by writing it to '1'.

10.3.4.3 PWM Mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the CCRx field of the TIMx_CCRx register.

$$\text{Period: Period} = (\text{ARR} + 1) * T_{CNT}$$

$$\text{Duty cycle: } D_{PWM} = \frac{CCR_x * T_{CNT}}{T_{PWM}}$$

$$\text{Where } T_{CNT} = T_{XTAL} * (\text{PSC} + 1)$$

The PWM mode can be selected independently on each channel (one PWM per OCx output) by setting '0' in the OCxM bits in the TIMx_CCRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCRx register, and the auto-reload preload register by setting the ARPE bit in the TIMx_CR register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable, using the CCxP bit in TIMx_CCRx register. It can be programmed as active high or active low.

In PWM mode, TIMx_CNT and CCRx (in TIMx_CCRx) are always compared to determine whether TIMx_CNT < CCRx. The PWM signal OCx is active as long as TIMx_CNT < CCRx, otherwise it becomes inactive. If the compare value in TIMx_CCRx is greater than the auto_reload value in TIMx_ARR, then PWM signal OCx output is active all the period. If the compare value is 0, then PWM signal OCx output is inactive all the period.

The timer is only able to generate PWM in edge-aligned mode. Fig 10-14 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

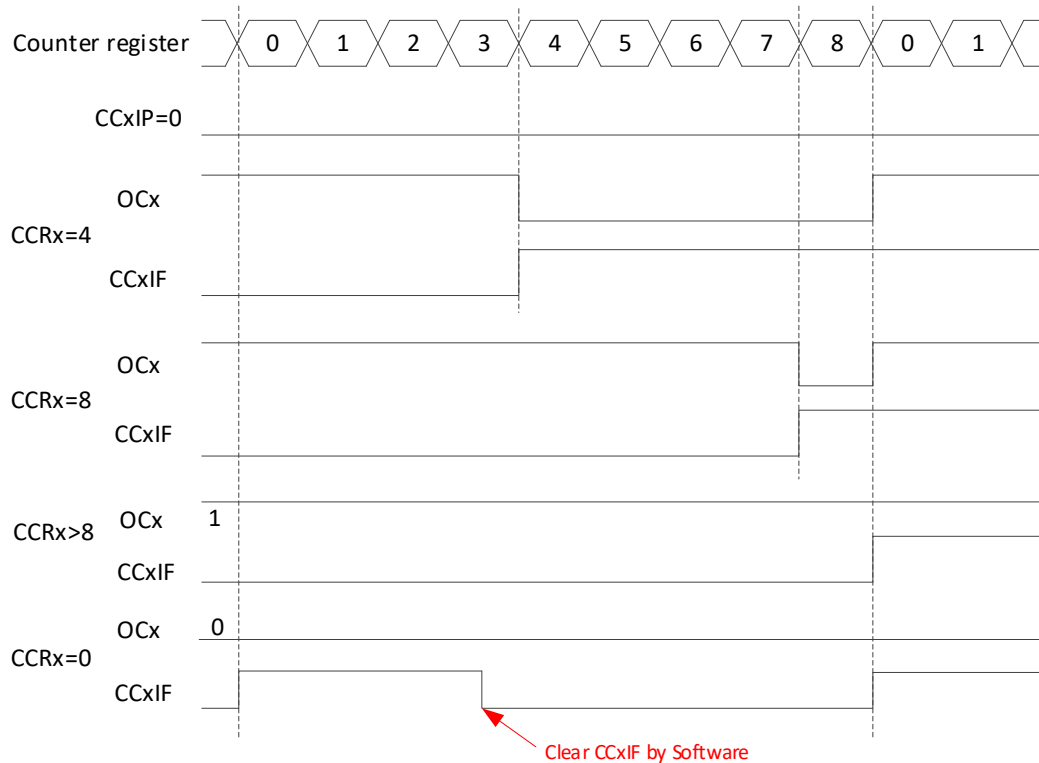


Fig 10-14 Edge-aligned PWM waveforms ($ARR=8$, $CCxP=0$)

10.3.4.4 One-pulse Mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the active edge of TRGI. Generating the waveform can be done in PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR register. This makes the counter stop automatically at the next update.

The one-pulse mode is shown in Fig 10-15.

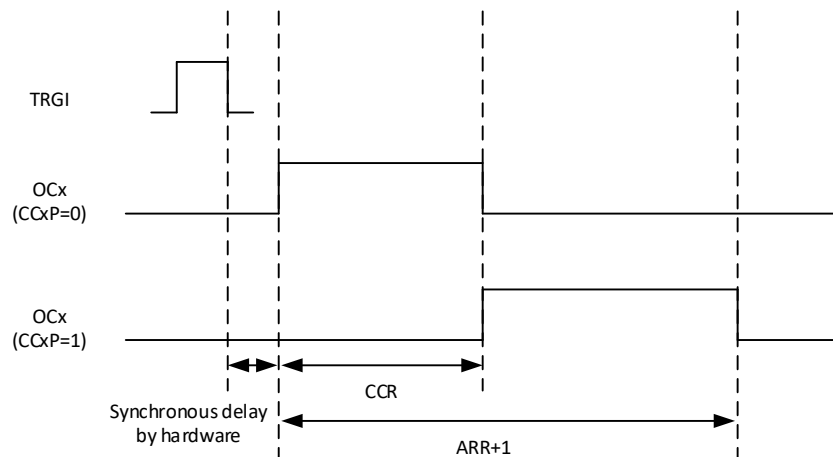


Fig 10-15 One-pulse mode timing

Note: Since the 18 channels are independent with each other, when TRGI is used to trigger one channel to output PWM signal, it can also be used as capture source in input capture mode of another channel.

10.4 Registers

The physical base address of all the timers is 0x4000_2000 in KM4 and 0x4800_2000 in KM0. Each timer has 128 bytes space for register setting.

Table 10-4 TIM0~TIM5 address table

Name	Address Offset
TIM0	0x000 ~ 0x07F
TIM1	0x080 ~ 0x0FF
TIM2	0x100 ~ 0x17F
TIM3	0x180 ~ 0x1FF
TIM4	0x200 ~ 0x27F
TIM5	0x280 ~ 0x2FF

10.4.1 TIM0/TIM1/TIM2/TIM3 Registers

The details of TIM0/TIM1/TIM2/TIM3 registers are listed in Table 10-5.

Table 10-5 TIM0/TIM1/TIM2/TIM3 memory map

Name	Address Offset	Access	Description
TIMx_EN	0x00	R/W	TIMx enable register
TIMx_CR	0x04	R/W	TIMx control register
TIMx_DIER	0x08	R/W	TIMx interrupt enable register
TIMx_SR	0x0C	R/W	TIMx status register
TIMx_EGR	0x10	W	TIMx event generation register
TIMx_CNT	0x14	R/W	TIMx counter register
RSVD	0x18	N/A	Reserved
TIMx_ARR	0x1C	R/W	TIMx auto-reload register
RSVD	0x20~0x7F	N/A	Reserved

10.4.1.1 TIMx Enable Register (TIMx_EN)

- **Name:** TIMx enable register (x = {0, 1, 2, 3})
- **Address offset:** 0x00
- **Reset value:** 0x00000000
- **Read/write access:** read/write
-

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															CNT_STS
															R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							CNT_RUN	RSVD						CNT_STOP	CNT_START
							R							W	W

Bit	Name	Access	Reset	Description
31:17	RSVD	N/A	-	Reserved
16	CNT_STS	R	0	Counter working status <ul style="list-style-type: none"> ● 0: Counter is stopped. ● 1: Counter is working.
15:9	RSVD	N/A	-	Reserved

8	CNT_RUN	R	0	Counter run status <ul style="list-style-type: none"> 0: Counter is disabled. 1: Counter is enabled.
7:2	RSVD	N/A	-	Reserved
1	CNT_STOP	W	0	Counter stop <ul style="list-style-type: none"> 0: No action. 1: Disable the counter. Poll CNT_RUN to see the counter status. If CNT_RUN is 0, it means that the counter has been disabled internally.
0	CNT_START	W	0	Counter start <ul style="list-style-type: none"> 0: No action. 1: Enable the counter. Poll CNT_RUN to see the counter status. If CNT_RUN is 1, it means that the counter has been enabled internally.

10.4.1.2 TIMx Control Register (TIMx_CR)

- Name:** TIMx control register (x = {0, 1, 2, 3})
- Address offset:** 0x04
- Reset value:** 0x00000000
- Read/write access:** read/write

31	30	29	...	7	6	5	4	3	2	1	0
RSVD							ARPE	RSVD	URS	UDIS	RSVD
							R/W		R/W	R/W	

Bit	Name	Access	Reset	Description
31:5	RSVD	N/A	-	Reserved
4	ARPE	R/W	0	Auto-reload preload enable <ul style="list-style-type: none"> 0: The TIMx_ARR register isn't buffered. 1: The TIMx_ARR register is buffered.
3	RSVD	N/A	-	Reserved
2	URS	R/W	0	Update request source <ul style="list-style-type: none"> 0: Update events can be <ul style="list-style-type: none"> Counter overflow Setting the UG bit 1: Counter overflow generates an update event.
1	UDIS	R/W	0	Update disable <ul style="list-style-type: none"> 0: UEV is enabled. Buffered registers are loaded with their preload values when UEV happens. 1: UEV is disabled. Shadow registers keep their value.
0	RSVD	N/A	-	Reserved

10.4.1.3 TIMx Interrupt Enable Register (TIMx_DIER)

- Name:** TIMx interrupt enable register (x = {0, 1, 2, 3})
- Address offset:** 0x08
- Reset value:** 0x00000000
- Read/write access:** read/write

31	30	29	28	11	...	5	4	3	2	1	0
RSVD											UIE
											R/W

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	UIE	R/W	0	Update interrupt enable <ul style="list-style-type: none"> 0: Update interrupt is disabled. 1: Update interrupt is enabled.

10.4.1.4 TIMx Status Register (TIMx_SR)

- **Name:** TIMx status register (x = {0, 1, 2, 3})
- **Address offset:** 0x0C
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	...	4	3	2	1	0
UG_DONE	RSVD									UIF
R										R/W1C

Bit	Name	Access	Reset	Description
31	UG_DONE	R	1	UG operation status This bit is cleared by hardware when the UG bit in the TIMx_EGR register is set. When the UG operation is done, hardware set this bit to '1'. So, software can poll this bit to see the UG operation status.
30:1	RSVD	N/A	-	Reserved
0	UIF	R/W1C	0	Update interrupt flag

10.4.1.5 TIMx Event Generation Register (TIMx_EGR)

- **Name:** TIMx event generation register (x = {0, 1, 2, 3})
- **Address offset:** 0x10
- **Reset value:** 0x00000000
- **Read/write access:** write

31	30	29	28	27	...	5	4	3	2	1	0
RSVD											UG
											W

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	UG	W	-	Update generation <ul style="list-style-type: none"> ● 0: No action. ● 1: Re-initialize the counter and generate an update of the registers. Note that the prescaler counter is cleared too, anyway the prescaler ratio isn't affected.

10.4.1.6 TIMx Counter Register (TIMx_CNT)

- **Name:** TIMx counter register (x = {0, 1, 2, 3})
- **Address offset:** 0x14
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	...	5	4	3	2	1	0
CNT												
R/W												

Bit	Name	Access	Reset	Description
31:0	CNT	R/W	0	Counter value

10.4.1.7 TIMx Auto-reload Register (TIMx_ARR)

- **Name:** TIMx auto-reload register (x = {0, 1, 2, 3})
- **Address offset:** 0x1C

- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	...	5	4	3	2	1	0
ARR												
R/W												

Bit	Name	Access	Reset	Description
31:0	ARR	R/W	0	ARR is the value to be loaded in the actual auto-reload register. It can be preloaded by setting the ARPE bit in the TIMx_CR register.

10.4.2 TIM4 Registers

The details of TIM4 registers are listed in Table 10-6.

Table 10-6 TIM4 memory map

Name	Address Offset	Access	Description
TIMx_EN	0x00	R/W	TIM4 enable register
TIMx_CR	0x04	R/W	TIM4 control register
TIMx_DIER	0x08	R/W	TIM4 interrupt enable register
TIMx_SR	0x0C	R/W	TIM4 status register
TIMx_EGR	0x10	W	TIM4 event generation register
TIMx_CNT	0x14	R/W	TIM4 counter register
TIMx_PSC	0x18	R/W	TIM4 prescaler register
TIMx_ARR	0x1C	R/W	TIM4 auto-reload register
TIMx_CCR0	0x20	R/W	TIM4 capture/compare register 0
RSVD	0x24 ~ 0x7F	N/A	Reserved

10.4.2.1 TIMx Enable Register (TIMx_EN)

- **Name:** TIM4 enable register
- **Address offset:** 0x00
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															CNT_STS
															R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							CNT_RUN	RSVD						CNT_STOP	CNT_START
							R							W	W

Bit	Name	Access	Reset	Description
31:17	RSVD	N/A	-	Reserved
16	CNT_STS	R	0	Counter working status <ul style="list-style-type: none"> ● 0: Counter is stopped ● 1: Counter is working
15:9	RSVD	N/A	-	Reserved
8	CNT_RUN	R	0	Counter run status <ul style="list-style-type: none"> ● 0: Counter is disabled ● 1: Counter is enabled
7:2	RSVD	N/A	-	Reserved
1	CNT_STOP	W	0	Counter stop <ul style="list-style-type: none"> ● 0: No action.

				<ul style="list-style-type: none"> 1: Disable the counter. Poll CNT_RUN to see the counter status. If CNT_RUN is 0, it means that the counter has been disabled internally
0	CNT_START	W	0	Counter start <ul style="list-style-type: none"> 0: No action. 1: Enable the counter. Poll CNT_RUN to see the counter status. If CNT_RUN is 1, it means that the counter has been enabled internally.

10.4.2.2 TIMx Control Register (TIMx_CR)

- **Name:** TIM4 control register
- **Address offset:** 0x04
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	...	8	7	6	5	4	3	2	1	0
RSVD									ARPE	RSVD	URS	UDIS	RSVD
									R/W		R/W	R/W	

Bit	Name	Access	Reset	Description
31:5	RSVD	N/A	-	Reserved
4	ARPE	R/W	0	Auto-reload preload enable <ul style="list-style-type: none"> 0: The TIMx_ARR register isn't buffered. 1: The TIMx_ARR register is buffered.
3	RSVD	N/A	-	Reserved
2	URS	R/W	0	Update request source <ul style="list-style-type: none"> 0: Update events can be <ul style="list-style-type: none"> Counter overflow Setting the UG bit 1: Counter overflow generates an update event.
1	UDIS	R/W	0	Update disable <ul style="list-style-type: none"> 0: UEV is enabled. Buffered registers are loaded with their preload values when UEV happens. 1: UEV is disabled. Shadow registers keep their value.
0	RSVD	N/A	-	Reserved

10.4.2.3 TIMx Interrupt Enable Register (TIMx_DIER)

- **Name:** TIM4 interrupt enable register
- **Address offset:** 0x08
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	...	5	4	3	2	1	0
RSVD									CCOIE	UIE
									R/W	R/W

Bit	Name	Access	Reset	Description
31:2	RSVD	N/A	-	Reserved
1	CCOIE	R/W	0	Capture/compare 0 interrupt enable <ul style="list-style-type: none"> 0: CCO interrupt is disabled. 1: CCO interrupt is enabled.
0	UIE	R/W	0	Update interrupt enable <ul style="list-style-type: none"> 0: Update interrupt is disabled. 1: Update interrupt is enabled.

10.4.2.4 TIMx Status Register (TIMx_SR)

- **Name:** TIM4 status register
- **Address offset:** 0x0C
- **Reset value:** 0x80000000
- **Read/write access:** read/write

31	30	29	28	27	...	5	4	3	2	1	0
UG_DONE	RSVD									CC0IF	UIF
R										R/W1C	R/W1C

Bit	Name	Access	Reset	Description
31	UG_DONE	R	1	UG operation status This bit is cleared by hardware when the UG bit in the TIMx_EGR register is set. When the UG operation is done, hardware set this bit to '1'. So, software can poll this bit to see the UG operation status.
30:2	RSVD	N/A	-	Reserved
1	CC0IF	R/W1C	0	Capture/compare 0 interrupt flag <ul style="list-style-type: none"> ● If channel CC0 is configured as pulse mode 0, this bit is set when TRGI is transferred to inactive level from active level. ● If channel CC0 is configured as pulse mode 1, this flag is set by hardware when the counter overflows. It is cleared by software.
0	UIF	R/W1C	0	Update interrupt flag.

10.4.2.5 TIMx Event Generation Register (TIMx_EGR)

- **Name:** TIM4 event generation register
- **Address offset:** 0x10
- **Reset value:** 0x00000000
- **Read/write access:** write

31	30	29	28	...	5	4	3	2	1	0
RSVD									CC0G	UG
									W	W

Bit	Name	Access	Reset	Description
31:2	RSVD	N/A	-	Reserved
1	CC0G	W	-	Capture/compare 1 generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. <ul style="list-style-type: none"> ● 0: No action. ● 1: A capture/compare event is generated on channel 1. The current value of the counter is captured in the TIMx_CCR0 register. The CC0IF flag is set, the corresponding interrupt is sent if enabled.
0	UG	W	-	Update generation <ul style="list-style-type: none"> ● 0: No action. ● 1: Re-initialize the counter and generate an update of the registers. Note that the prescaler counter is cleared too, anyway the prescaler ratio isn't affected.

10.4.2.6 TIMx Counter Register (TIMx_CNT)

- **Name:** TIM4 counter register
- **Address offset:** 0x14
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
R/W															

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	CNT	R/W	0	Counter value

10.4.2.7 TIMx Prescaler Register (TIMx_PSC)

- **Name:** TIM4 prescaler register
- **Address offset:** 0x18
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	...	12	11	10	9	8	7	6	...	1	0
RSVD												PSC			
												R/W			

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7:0	PSC	R/W	0	Prescaler value The counter clock frequency is equal to $f_{CK_PSC} / (PSC[7:0] + 1)$. PSC contains the value to be loaded in the actual prescaler register at each update event, including when the counter is cleared through the UG bit in the TIMx_EGR register.

10.4.2.8 TIMx Auto-reload Register (TIMx_ARR)

- **Name:** TIM4 auto-reload register
- **Address offset:** 0x1C
- **Reset value:** 0x0000FFFF
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR															
R/W															

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	ARR	R/W	0	ARR is the value to be loaded in the actual auto-reload register. It can be preloaded by setting the ARPE bit in the TIMx_CR register.

10.4.2.9 TIMx Capture/Compare Register 0 (TIMx_CCR0)

- **Name:** TIM4 capture/compare register 0
- **Address offset:** 0x20

- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD			CCOM	RSVD	CCOP	RSVD	CCOE	RSVD							
			R/W		R/W		R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRO															
R/W															

Bit	Name	Access	Reset	Description
31:29	RSVD	N/A	-	Reserved
28	CCOM	R/W	0	CC0 pulse mode <ul style="list-style-type: none"> ● 0: Pulse mode 0 ● 1: Pulse mode 1
27	RSVD	N/A	-	Reserved
26	CCOP	R/W	0	CC0 channel configured as input <ul style="list-style-type: none"> ● 0: Positive edge of TRGI is active for capture. ● 1: Negative edge of TRGI is active for capture.
25	RSVD	N/A	-	Reserved
24	CCOE	R/W	0	CC0 enable <ul style="list-style-type: none"> ● 0: CC0 is disabled ● 1: CC0 is enabled
23:16	RSVD	N/A	-	Reserved
15:0	CCRO	R/W	0	Capture/compare 0 value <ul style="list-style-type: none"> ● If channel CC0 is configured as pulse mode 0: CCRO is the pulse width of input capture event (TRGI). ● If channel CC0 is configured as pulse mode 1: CCRO is the pulse number of input capture event (TRGI) when the counter counts from 0 to ARR.

10.4.3 TIM5 Registers

The details of TIM5 registers are listed in Table 10-7.

Table 10-7 TIM5 memory map

Name	Address Offset	Access	Description
TIMx_EN	0x00	R/W	TIM5 enable register
TIMx_CR	0x04	R/W	TIM5 control register
TIMx_DIER	0x08	R/W	TIM5 interrupt enable register
TIMx_SR	0x0C	R/W	TIM5 status register
TIMx_EGR	0x10	W	TIM5 event generation register
TIMx_CNT	0x14	R/W	TIM5 counter register
TIMx_PSC	0x18	R/W	TIM5 prescaler register
TIMx_ARR	0x1C	R/W	TIM5 auto-reload register
TIMx_CCR0	0x20	R/W	TIM5 capture/compare register 0
TIMx_CCR1	0x24	R/W	TIM5 capture/compare register 1
TIMx_CCR2	0x28	R/W	TIM5 capture/compare register 2
TIMx_CCR3	0x2C	R/W	TIM5 capture/compare register 3
TIMx_CCR4	0x30	R/W	TIM5 capture/compare register 4
TIMx_CCR5	0x34	R/W	TIM5 capture/compare register 5
TIMx_CCR6	0x38	R/W	TIM5 capture/compare register 6
TIMx_CCR7	0x3C	R/W	TIM5 capture/compare register 7
TIMx_CCR8	0x40	R/W	TIM5 capture/compare register 8
TIMx_CCR9	0x44	R/W	TIM5 capture/compare register 9
TIMx_CCR10	0x48	R/W	TIM5 capture/compare register 10

TIMx_CCR11	0x4C	R/W	TIM5 capture/compare register 11
TIMx_CCR12	0x50	R/W	TIM5 capture/compare register 12
TIMx_CCR13	0x54	R/W	TIM5 capture/compare register 13
TIMx_CCR14	0x58	R/W	TIM5 capture/compare register 14
TIMx_CCR15	0x5C	R/W	TIM5 capture/compare register 15
TIMx_CCR16	0x60	R/W	TIM5 capture/compare register 16
TIMx_CCR17	0x64	R/W	TIM5 capture/compare register 17
RSVD	0x68 ~ 0x7F	-	Reserved

10.4.3.1 TIMx Enable Register (TIMx_EN)

- **Name:** TIM5 enable register
- **Address offset:** 0x00
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															CNT_STS
															R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								CNT_RUN	RSVD					CNT_STOP	CNT_START
								R						W	W

Bit	Name	Access	Reset	Description
31:17	RSVD	N/A	-	Reserved
16	CNT_STS	R	0	Counter working status <ul style="list-style-type: none"> ● 0: Counter is stopped ● 1: Counter is working
15:9	RSVD	N/A	-	Reserved
8	CNT_RUN	R	0	Counter run status <ul style="list-style-type: none"> ● 0: Counter is disabled ● 1: Counter is enabled
7:2	RSVD	N/A	-	Reserved
1	CNT_STOP	W	0	Counter stop <ul style="list-style-type: none"> ● 0: No action. ● 1: Disable the counter. Poll CNT_RUN to see the counter status. If CNT_RUN is 0, it means that the counter has been disabled internally
0	CNT_START	W	0	Counter start <ul style="list-style-type: none"> ● 0: No action. ● 1: Enable the counter. Poll CNT_RUN to see the counter status. If CNT_RUN is 1, it means that the counter has been enabled internally.

10.4.3.2 TIMx Control Register (TIMx_CR)

- **Name:** TIM5 control register
- **Address offset:** 0x04
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	...	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							ETP	RSVD			ARPE	OPM	URS	UDIS	RSVD
							R/W				R/W	R/W	R/W	R/W	

Bit	Name	Access	Reset	Description
31:9	RSVD	N/A	-	Reserved

8	ETP	R/W	0	External trigger polarity (TRGI). ● 0: Positive edge is active. ● 1: Negative edge is active. Note: This bit is only valid in one-pulse mode.
7:5	RSVD	N/A	-	Reserved
4	ARPE	R/W	0	Auto-reload preload enable ● 0: The TIMx_ARR register isn't buffered. ● 1: The TIMx_ARR register is buffered.
3	OPM	R/W	0	One pulse mode ● 0: Counter isn't stopped at update event. ● 1: Counter stops counting at the next update event.
2	URS	R/W	0	Update request source ● 0: Update events can be ■ Counter overflow ■ Setting the UG bit ● 1: Counter overflow generates an update event.
1	UDIS	R/W	0	Update disable ● 0: UEV is enabled. Buffered registers are loaded with their preload values when UEV happens. ● 1: UEV is disabled. Shadow registers keep their value.
0	RSVD	N/A	-	Reserved

10.4.3.3 TIMx Interrupt Enable Register (TIMx_DIER)

- **Name:** TIM5 interrupt enable register
- **Address offset:** 0x08
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD													CC17IE	CC16IE	CC15IE
													R/W	R/W	R/W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC14IE	CC13IE	CC12IE	CC11IE	CC10IE	CC9IE	CC8IE	CC7IE	CC6IE	CC5IE	CC4IE	CC3IE	CC2IE	CC1IE	CC0IE	UIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:19	RSVD	N/A	-	Reserved
18	CC17IE	R/W	0	Capture/compare 17 interrupt enable ● 0: CC17 interrupt is disabled ● 1: CC17 interrupt is enabled
17	CC16IE	R/W	0	Capture/compare 16 interrupt enable ● 0: CC16 interrupt is disabled ● 1: CC16 interrupt is enabled
16	CC15IE	R/W	0	Capture/compare 15 interrupt enable ● 0: CC15 interrupt is disabled ● 1: CC15 interrupt is enabled
15	CC14IE	R/W	0	Capture/compare 14 interrupt enable ● 0: CC14 interrupt is disabled ● 1: CC14 interrupt is enabled
14	CC13IE	R/W	0	Capture/compare 13 interrupt enable ● 0: CC13 interrupt is disabled ● 1: CC13 interrupt is enabled
13	CC12IE	R/W	0	Capture/compare 12 interrupt enable ● 0: CC12 interrupt is disabled ● 1: CC12 interrupt is enabled

12	CC11IE	R/W	0	Capture/compare 11 interrupt enable ● 0: CC11 interrupt is disabled ● 1: CC11 interrupt is enabled
11	CC10IE	R/W	0	Capture/compare 10 interrupt enable ● 0: CC10 interrupt is disabled ● 1: CC10 interrupt is enabled
10	CC9IE	R/W	0	Capture/compare 9 interrupt enable ● 0: CC9 interrupt is disabled ● 1: CC9 interrupt is enabled
9	CC8IE	R/W	0	Capture/compare 8 interrupt enable ● 0: CC8 interrupt is disabled ● 1: CC8 interrupt is enabled
8	CC7IE	R/W	0	Capture/compare 7 interrupt enable ● 0: CC7 interrupt is disabled ● 1: CC7 interrupt is enabled
7	CC6IE	R/W	0	Capture/compare 6 interrupt enable ● 0: CC6 interrupt is disabled ● 1: CC6 interrupt is enabled
6	CC5IE	R/W	0	Capture/compare 5 interrupt enable ● 0: CC5 interrupt is disabled ● 1: CC5 interrupt is enabled
5	CC4IE	R/W	0	Capture/compare 4 interrupt enable ● 0: CC4 interrupt is disabled ● 1: CC4 interrupt is enabled
4	CC3IE	R/W	0	Capture/compare 3 interrupt enable ● 0: CC3 interrupt is disabled ● 1: CC3 interrupt is enabled
3	CC2IE	R/W	0	Capture/compare 2 interrupt enable ● 0: CC2 interrupt is disabled ● 1: CC2 interrupt is enabled
2	CC1IE	R/W	0	Capture/compare 1 interrupt enable ● 0: CC1 interrupt is disabled ● 1: CC1 interrupt is enabled
1	CC0IE	R/W	0	Capture/compare 0 interrupt enable ● 0: CC0 interrupt is disabled ● 1: CC0 interrupt is enabled
0	UIE	R/W	0	Update interrupt enable ● 0: Update interrupt is disabled ● 1: Update interrupt is enabled

10.4.3.4 TIMx Status Register (TIMx_SR)

- **Name:** TIM5 status register
- **Address offset:** 0x0C
- **Reset value:** 0x80000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UG_DON E	RSVD												CC18IF	CC17IF	CC16IF
R													R/W1 C	R/W1 C	R/W1 C
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC15IF	CC14IF	CC13IF	CC12IF	CC11IF	CC10IF	CC9IF	CC8IF	CC7IF	CC6IF	CC5IF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
R/W1C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C	R/W1 C

Bit	Name	Access	Reset	Description
31	UG_DONE	R	1	UG operation status This bit is cleared by hardware when the UG bit in the TIMx_EGR register is set. When the UG operation is done, hardware set this bit to '1'. So, software can poll this bit to see the UG operation status.
30:19	RSVD	N/A	-	Reserved
18	CC17IF	R/W1C	0	Refer to CC0IF description
17	CC16IF	R/W1C	0	Refer to CC0IF description
16	CC15IF	R/W1C	0	Refer to CC0IF description
15	CC14IF	R/W1C	0	Refer to CC0IF description
14	CC13IF	R/W1C	0	Refer to CC0IF description
13	CC12IF	R/W1C	0	Refer to CC0IF description
12	CC11IF	R/W1C	0	Refer to CC0IF description
11	CC10IF	R/W1C	0	Refer to CC0IF description
10	CC9IF	R/W1C	0	Refer to CC0IF description
9	CC8IF	R/W1C	0	Refer to CC0IF description
8	CC7IF	R/W1C	0	Refer to CC0IF description
7	CC6IF	R/W1C	0	Refer to CC0IF description
6	CC5IF	R/W1C	0	Refer to CC0IF description
5	CC4IF	R/W1C	0	Refer to CC0IF description
4	CC3IF	R/W1C	0	Refer to CC0IF description
3	CC2IF	R/W1C	0	Refer to CC0IF description
2	CC1IF	R/W1C	0	Refer to CC0IF description
1	CC0IF	R/W1C	0	Capture/compare 0 interrupt flag <ul style="list-style-type: none"> If channel CC0 is configured as output: This flag is set by hardware when the counter matches the compare value. It is cleared by software. <ul style="list-style-type: none"> 0: No match. 1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR0 register. When the contents of TIMx_CCR0 are greater than the contents of the TIMx_ARR, the CC0IF bit goes high on the counter overflow. If channel CC0 is configured as input: This bit is set by hardware on a capture. It is cleared by software. <ul style="list-style-type: none"> 0: No input capture is occurred. 1: The counter value has been captured in the CCR0 field of the TIMx_CCR0 register. An active edge has been detected.
0	UIF	R/W1C	0	Update interrupt flag

10.4.3.5 TIMx Event Generation Register (TIMx_EGR)

- Name:** TIM5 event generation register
- Address offset:** 0x10
- Reset value:** 0x00000000
- Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD													CC17G	CC16G	CC15G
													W	W	W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC14G	CC13G	CC12G	CC11G	CC10G	CC9G	CC8G	CC7G	CC6G	CC5G	CC4G	CC3G	CC2G	CC1G	CC0G	UG
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Bit	Name	Access	Reset	Description
31:19	RSVD	N/A	-	Reserved
18	CC17G	W	0	Refer to CC0G description
17	CC16G	W	0	Refer to CC0G description
16	CC15G	W	0	Refer to CC0G description

15	CC14G	W	0	Refer to CC0G description
14	CC13G	W	0	Refer to CC0G description
13	CC12G	W	0	Refer to CC0G description
12	CC11G	W	0	Refer to CC0G description
11	CC10G	W	0	Refer to CC0G description
10	CC9G	W	0	Refer to CC0G description
9	CC8G	W	0	Refer to CC0G description
8	CC7G	W	0	Refer to CC0G description
7	CC6G	W	0	Refer to CC0G description
6	CC5G	W	0	Refer to CC0G description
5	CC4G	W	0	Refer to CC0G description
4	CC3G	W	0	Refer to CC0G description
3	CC2G	W	0	Refer to CC0G description
2	CC1G	W	0	Refer to CC0G description
1	CC0G	W	0	Capture/compare 0 generation <ul style="list-style-type: none"> This bit is set by software in order to generate an event. It is automatically cleared by hardware. 0: No action. 1: A capture/compare event is generated on channel 0. <ul style="list-style-type: none"> If channel CC0 is configured as output: CC0IF flag is set, corresponding interrupt is sent if enabled. If channel CC0 is configured as input: The current value of the counter is captured in the CCR0 field of the TIMx_CCR0 register. The CC0IF flag is set, the corresponding interrupt is sent if enabled.
0	UG	W	0	Update generation <ul style="list-style-type: none"> 0: No action. 1: Re-initialize the counter and generate an update of the registers. Note that the prescaler counter is cleared too, anyway the prescaler ratio isn't affected.

10.4.3.6 TIMx Counter Register (TIMx_CNT)

- **Name:** TIM5 counter register
- **Address offset:** 0x14
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
R/W															

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	CNT	R/W	0	Counter value

10.4.3.7 TIMx Prescaler Register (TIMx_PSC)

- **Name:** TIM5 prescaler register
- **Address offset:** 0x18
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	...	12	11	10	9	8	7	6	...	1	0
RSVD											PSC				
											R/W				

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7:0	PSC	R/W	0	Prescaler value The counter clock frequency is equal to $f_{CK_PSC} / (PSC + 1)$. PSC contains the value to be loaded in the actual prescaler register at each update event, including when the counter is cleared through the UG bit of the TIMx_EGR register.

10.4.3.8 TIMx Auto-reload Register (TIMx_ARR)

- **Name:** TIM5 auto-reload register
- **Address offset:** 0x1C
- **Reset value:** 0x0000FFFF
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR															
R/W															

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	ARR	R/W	0	ARR is the value to be loaded in the actual auto-reload register. It can be preloaded by setting the ARPE bit in the TIMx_CR register.

10.4.3.9 TIMx Capture/Compare Register 0 (TIMx_CCR0)

- **Name:** TIM5 capture/compare register 0
- **Address offset:** 0x20
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CCOM	CCOP	OCOPE	CCOE	RSVD							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRO															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CCOM	R/W	0	CC0 working mode <ul style="list-style-type: none"> ● 0: PWM mode ● 1: Input capture mode
26	CCOP	R/W	0	<ul style="list-style-type: none"> ● CC0 channel configured as output <ul style="list-style-type: none"> ■ 0: OC0 active is high. ■ 1: OC0 active is low. ● CC0 channel configured as input <ul style="list-style-type: none"> ■ 0: Positive edge of TRGI is active for capture.

				<p>■ 1: Negative edge of TRGI is active for capture.</p>
25	OCOPE	R/W	0	<p>Output Compare 0 preload enable</p> <ul style="list-style-type: none"> 0: Preload register on CCR0 is disabled. CCR0 can be written at any time, the new value is taken in account immediately. 1: Preload register on CCR0 is enabled. Read/write operations access the preload register. CCR0 preload value is loaded in the active register at each update event.
24	CCOE	R/W	0	<p>CC0 enable</p> <ul style="list-style-type: none"> 0: CC0 is disabled. 1: CC0 is enabled.
23:16	RSVD	N/A	-	Reserved
15:0	CCR0	R/W	0	<p>Capture/compare 0 value</p> <ul style="list-style-type: none"> If channel CC0 is configured as output: CCR0 is the value to be loaded in the actual capture/compare 0 register (preload value). It is loaded permanently if the preload feature isn't selected in the OCOPE bit. Else the preload value is copied in the active capture/compare 0 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC0 output. If channel CC0 is configured as input: CCR0 is the counter value transferred by the last input capture event (TRGI). <p>Note: Value must be 0 ~ 100 (including 0 & 100).</p>

10.4.3.10 TIMx Capture/Compare Register 1 (TIMx_CCR1)

- **Name:** TIM5 capture/compare register 1
- **Address offset:** 0x24
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC1M	CC1P	OC1PE	CC1E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC1M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC1P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC1PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC1E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR1	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.11 TIMx Capture/Compare Register 2 (TIMx_CCR2)

- **Name:** TIM5 capture/compare register 2
- **Address offset:** 0x28
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC2M	CC2P	OC2PE	CC2E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															

R/W

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC2M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC2P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC2PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC2E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR2	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.12 TIMx Capture/Compare Register 3 (TIMx_CCR3)

- **Name:** TIM5 capture/compare register 3
- **Address offset:** 0x2C
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC3M	CC3P	OC3PE	CC3E	RSVD							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC3M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC3P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC3PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC3E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR3	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.13 TIMx Capture/Compare Register 4 (TIMx_CCR4)

- **Name:** TIM5 capture/compare register 4
- **Address offset:** 0x30
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC4M	CC4P	OC4PE	CC4E	RSVD							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC4M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC4P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC4PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC4E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved

15:0	CCR4	R/W	0	Refer to CCR0 description in TIMx_CCR0
------	------	-----	---	--

10.4.3.14 TIMx Capture/Compare Register 5 (TIMx_CCR5)

- **Name:** TIM5 capture/compare register 5
- **Address offset:** 0x34
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC5M	CC5P	OC5PE	CC5E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC5M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC5P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC5PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC5E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR5	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.15 TIMx Capture/Compare Register 6 (TIMx_CCR6)

- **Name:** TIM5 capture/compare register 6
- **Address offset:** 0x38
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC6M	CC6P	OC6PE	CC6E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC6M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC6P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC6PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC6E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR6	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.16 TIMx Capture/Compare Register 7 (TIMx_CCR7)

- **Name:** TIM5 capture/compare register 7
- **Address offset:** 0x3C
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC7M	CC7P	OC7PE	CC7E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR7															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC7M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC7P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC7PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC7E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR7	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.17 TIMx Capture/Compare Register 8 (TIMx_CCR8)

- **Name:** TIM5 capture/compare register 8
- **Address offset:** 0x40
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC8M	CC8P	OC8PE	CC8E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR8															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC8M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC8P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC8PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC8E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR8	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.18 TIMx Capture/Compare Register 9 (TIMx_CCR9)

- **Name:** TIM5 capture/compare register 9
- **Address offset:** 0x44
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC9M	CC9P	OC9PE	CC9E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR9															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved

27	CC9M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC9P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC9PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC9E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR9	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.19 TIMx Capture/Compare Register 10 (TIMx_CCR10)

- **Name:** TIM5 capture/compare register 10
- **Address offset:** 0x48
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC10M	CC10P	OC10PE	CC10E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR10															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC10M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC10P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC10PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC10E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR10	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.20 TIMx Capture/Compare Register 11 (TIMx_CCR11)

- **Name:** TIM5 capture/compare register 11
- **Address offset:** 0x4C
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC11M	CC11P	OC11PE	CC11E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR11															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC11M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC11P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC11PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC11E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR11	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.21 TIMx Capture/Compare Register 12 (TIMx_CCR12)

- **Name:** TIM5 capture/compare register 12
- **Address offset:** 0x50
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC12M	CC12P	OC12PE	CC12E	RSVD							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR12															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC12M	R/W	0	Refer to CCOM description in TIMx_CCR0
26	CC12P	R/W	0	Refer to CCOP description in TIMx_CCR0
25	OC12PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC12E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR12	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.22 TIMx Capture/Compare Register 13 (TIMx_CCR13)

- **Name:** TIM5 capture/compare register 13
- **Address offset:** 0x54
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC13M	CC13P	OC13PE	CC13E	RSVD							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR13															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC13M	R/W	0	Refer to CCOM description in TIMx_CCR0
26	CC13P	R/W	0	Refer to CCOP description in TIMx_CCR0
25	OC13PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC13E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR13	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.23 TIMx Capture/Compare Register 14 (TIMx_CCR14)

- **Name:** TIM5 capture/compare register 14
- **Address offset:** 0x58
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC14M	CC14P	OC14PE	CC14E	RSVD							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR14															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC14M	R/W	0	Refer to CCOM description in TIMx_CCR0
26	CC14P	R/W	0	Refer to CCOP description in TIMx_CCR0
25	OC14PE	R/W	0	Refer to OCOPE description in TIMx_CCR0
24	CC14E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR14	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.24 TIMx Capture/Compare Register 15 (TIMx_CCR15)

- **Name:** TIM5 capture/compare register 15
- **Address offset:** 0x5C
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC15M	CC15P	OC15PE	CC15E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR15															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC15M	R/W	0	Refer to CCOM description in TIMx_CCR0
26	CC15P	R/W	0	Refer to CCOP description in TIMx_CCR0
25	OC15PE	R/W	0	Refer to OCOPE description in TIMx_CCR0
24	CC15E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR15	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.25 TIMx Capture/Compare Register 16 (TIMx_CCR16)

- **Name:** TIM5 capture/compare register 16
- **Address offset:** 0x60
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC16M	CC16P	OC16PE	CC16E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR16															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC16M	R/W	0	Refer to CCOM description in TIMx_CCR0
26	CC16P	R/W	0	Refer to CCOP description in TIMx_CCR0

25	OC16PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC16E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR16	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.4.3.26 TIMx Capture/Compare Register 17 (TIMx_CCR17)

- **Name:** TIM5 capture/compare register 17
- **Address offset:** 0x64
- **Reset value:** 0x00000000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CC17M	CC17P	OC17PE	CC17E	RSVD							
				R/W	R/W	R/W	R/W								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR17															
R/W															

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27	CC17M	R/W	0	Refer to CC0M description in TIMx_CCR0
26	CC17P	R/W	0	Refer to CC0P description in TIMx_CCR0
25	OC17PE	R/W	0	Refer to OC0PE description in TIMx_CCR0
24	CC17E	R/W	0	Refer to CC0E description in TIMx_CCR0
23:16	RSVD	N/A	-	Reserved
15:0	CCR17	R/W	0	Refer to CCR0 description in TIMx_CCR0

10.5 Design Implementation

10.5.1 Introduction

The Timer IP is connected under APB buses, which have one PWM timer, one pulse timer and four basic timers. The total block diagram of this IP is shown as Fig 10-16.

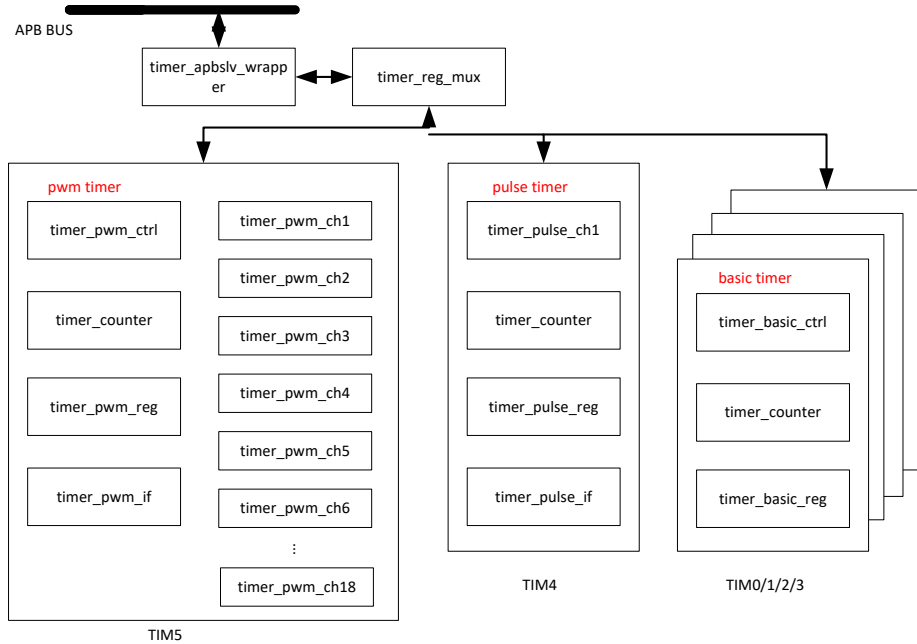


Fig 10-16 Block diagram

10.5.2 Synchronous Data from Fast Clock to Slow Clock

The TIMx_ARR can be configured dynamically, it is hard for basic timer to get the correct value of TIMx_ARR since the timer clock of basic timer is much slower than bus clock. You must latch a safe version of TIMx_ARR for basic timer clock to read.

The cki is fast clock and cko is slow clock. The data_i is data in fast clock domain, which can be changed at any time, and the data_o is the safe version of data_i for slow clock to read.

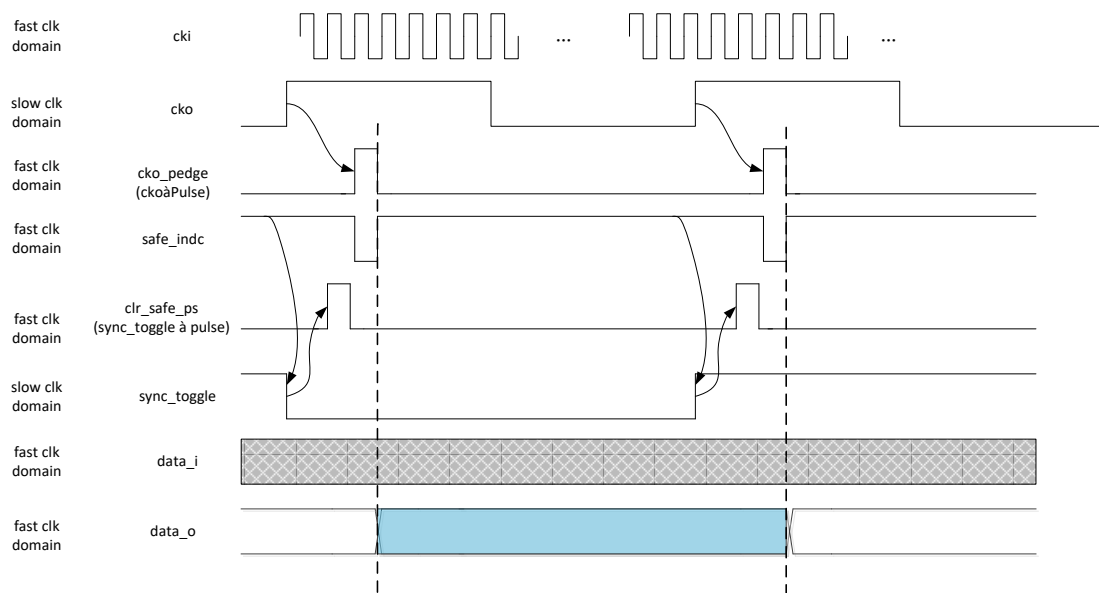


Fig 10-17 Synchronous data diagram

10.6 Operation Flow

10.6.1 Upcounting Mode

Upcounting mode: support TIM0~TIM5. The configuration flow is listed in Table 10-8.

Table 10-8 Timers upcounting configuration flow

Step	What to do	How to do	Comments
1	Check the timer run status	Poll CNT_RUN in TIMx_EN. If CNT_RUN is 1, go to step 2, else go to step 4	
2	Disable the timer	Write "0x02" to TIMx_EN	Stop Counter
3	Check the timer run status	Poll CNT_RUN in TIMx_EN until CNT_RUN is 0	Wait counter to be stopped
4	Set prescaler	Configure TIMx_PSC	
5	Set ARR	Configure TIMx_ARR	
6	Initialize the counter	Write "0x01" to TIMx_EGR (set UG bit)	Generate UEV by software
7	Check whether the timer is initialized	Poll UG_DONE in TIMx_SR until it is set	Wait counter to be initialized
8	Clear event flag	Write "0x0F" to TIMx_SR	Clear all flags
10	Enable the timer	Set CEN bit in TIMx_CR	Configure UEV condition at the same time
11	Check whether the timer is running	Poll CNT_RUN in TIMx_EN	Optional
12	Change ARR on the fly	Configure TIMx_ARR	Recommend to set ARPE bit in TIMx_CR

10.6.2 Pulse Mode

Pulse mode: support TIM4.

10.6.2.1 Pulse Mode 0 (Pulse Width)

The pulse mode 0 configuration flow is shown in Table 10-9.

Table 10-9 Pulse mode 0 configuration flow

Step	What to do	How to do	Comments
1	Disable the timer	Write "0x00" to TIMx_CR	Stop Counter
2	Set prescaler	Configure TIMx_PSC	
3	Set ARR	Configure TIMx_ARR	Must great than the width of TRGI
4	Configure pulse mode 0	Configure CCxPM bit in TIMx_CCRx Configure the pulse polarity of TRGI (CCxP in TIMx_CCRx)	
5	Initialize the counter	Write "0x01" to TIMx_EGR (set UG bit)	Generate UEV by software
6	Clear event flag	Write "0x0F" to TIMx_SR	Clear all flags
7	Enable the timer	Set CEN bit in TIMx_CR	Configure UEV condition at the same time
8	Read the current value of the counter when CCxIF is asserted	Read TIMx_CCRx	Use interrupt to notify CPU, must enable CCxIE in TIMx_DIER
9	Clear CCxIF	Write TIMx_SR	

The statistic process is repeated unless OPM bit is set in TIMx_CR.

10.6.2.2 Pulse Mode 1 (Pulse Number)

The pulse mode 1 configuration flow is shown in Table 10-10.

Table 10-10 Pulse mode 1 configuration flow

Step	What to do	How to do	Comments
1	Disable the timer	Write "0x00" to TIMx_CR	Stop Counter

2	Set prescaler	Configure TIMx_PSC	
3	Set ARR	Configure TIMx_ARR	Set the statistic period
4	Configure pulse mode 1	Configure CCxPM bit in TIMx_CCRx Configure the edge polarity of TRGI (CCxP in TIMx_CCRx)	
5	Initialize the counter	Write "0x01" to TIMx_EGR (set UG bit)	Generate UEV by software
6	Clear event flag	Write "0x0F" to TIMx_SR	Clear all flags
7	Enable the timer	Set CEN bit in TIMx_CR	Configure UEV condition at the same time
8	Read the pulse number when UIF is asserted	Read TIMx_CCRx	Use interrupt to notify CPU, must enable UIE in TIMx_DIER
9	Clear UIF	Write TIMx_SR	

The statistic process is repeated unless OPM bit is set in TIMx_CR.

10.6.3 PWM Mode

PWM mode: support TIM5.

10.6.3.1 Repeated Mode

The PWM repeated mode configuration flow is illustrated in Table 10-11.

Table 10-11 PWM repeated mode configuration flow

Step	What to do	How to do	Comments
1	Disable the timer	Write "0x00" to TIMx_CR	Stop Counter
2	Set prescaler	Configure TIMx_PSC	
3	Set ARR	Configure TIMx_ARR	
4	Configure PWM mode	Configure CCxM bit in TIMx_CCRx Configure the level polarity of OCx (CCxP in TIMx_CCRx)	
5	Set CCRx	Configure CCRx in TIMx_CCRx	
6	Initialize the counter	Write "0x01" to TIMx_EGR (set UG bit)	Generate UEV by software
7	Clear event flag	Write "0x0F" to TIMx_SR	Clear all flags
8	Enable the timer	Set CEN bit in TIMx_CR	Configure UEV condition at the same time
9	Change ARR on-the-fly	Configure TIMx_ARR	Recommend to set ARPE bit in TIMx_CR
10	Change CCRx on-the-fly	Configure CCRx in TIMx_CCRx	Recommend to set OCxPE bit in TIMx_CCRx

10.6.3.2 One-pulse Mode

The TIM5 one-pulse mode configuration flow is illustrated in Table 10-12.

Table 10-12 TIM5 one-pulse mode configuration flow

Step	What to do	How to do	Comments
1	Disable the timer	Write "0x00" to TIMx_CR	Stop Counter
2	Set prescaler	Configure TIMx_PSC	
3	Set ARR	Configure TIMx_ARR	
4	Configure PWM mode	Configure CCxM bit in TIMx_CCRx Configure the edge polarity of OCx (CCxP in TIMx_CCRx)	
5	Set CCRx	Configure CCRx in TIMx_CCRx	
6	Initialize the counter	Write "0x01" to TIMx_EGR (set UG bit)	Generate UEV by software
7	Clear event flag	Write "0x0F" to TIMx_SR	Clear all flags
8	Configure the active edge of input and enable the timer	Configure ETP Set CEN bit in TIMx_CR	Configure UEV condition at the same time

10.6.4 Input Capture Mode

Input capture mode: support TIM5. The TIM5 input capture mode configuration flow is illustrated in Table 10-13.

Table 10-13 TIM5 input capture mode configuration flow

Step	What to do	How to do	Comments
1	Disable the timer	Write "0x00" to TIMx_CR	Stop Counter
2	Set prescaler	Configure TIMx_PSC	
3	Set ARR	Configure TIMx_ARR	
4	Configure input capture mode	Configure CCxM bit in TIMx_CCRx Configure the polarity of TRGI (CCxP in TIMx_CCRx)	
5	Initial the counter	Write "0x01" to TIMx_EGR (set UG bit)	Generate UEV by software
6	Clear event flag	Write "0x0F" to TIMx_SR	Clear all flags
7	Enable the timer	Set CEN bit in TIMx_CR	Configure UEV condition at the same time
8	Read the current value of the counter when CCxIF is asserted	Read CCRx in TIMx_CCRx	Use interrupt to notify CPU, must enable CCxIE in TIMx_DIER
9	Clear CCxIF	Write TIMx_SR	

In input capture mode, different channels can be used to capture the double edge of TRGI, thus to get the width of TRGI.

11 Real-time Clock (RTC)

11.1 Product Overview

11.1.1 Introduction

The real-time clock (RTC) is an independent binary coded decimal (BCD) timer/counter. One 32-bit register contains the seconds, minutes, hours (12- or 24-hour format) expressed in BCD format. One 32-bit register contains the days expressed in binary format.

Daylight saving time compensation can be performed.

Additional two 32-bit registers contain the programmable alarm seconds, minutes, hours and days.

A digital calibration feature is available to compensate for some deviation.

After backup domain reset, all RTC registers are protected against possible parasitic write accesses.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status, that is run mode, low power mode or under reset.

11.1.2 Features

- Time with seconds, minutes, hours (12- or 24-hour format) and days
- Daylight saving compensation programmable by software
- One programmable alarm with interrupt function. The alarm can be triggered by any combination of the time fields.
- Maskable interrupt/event:
 - Alarm
- Digital calibration circuit
- Register write protection

11.1.3 Block Diagram

The RTC is connected under APB buses. The total block diagram is shown in Fig 11-1.

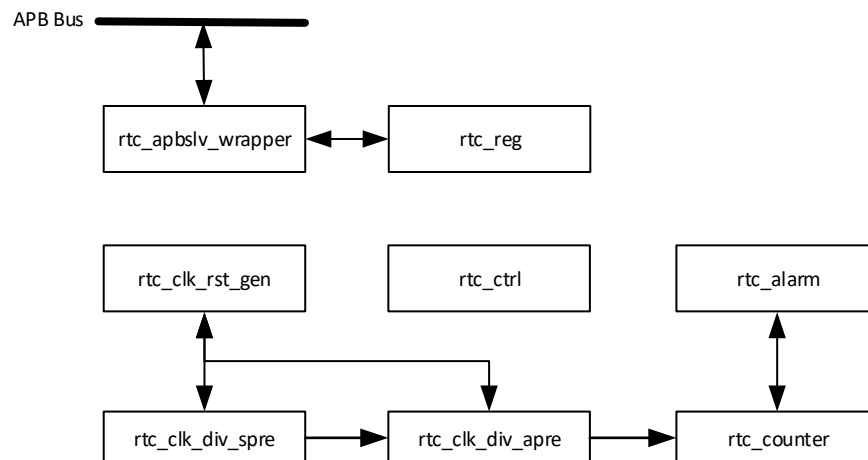


Fig 11-1 RTC block diagram

The RTC prescale diagram is shown in Fig 11-2 .

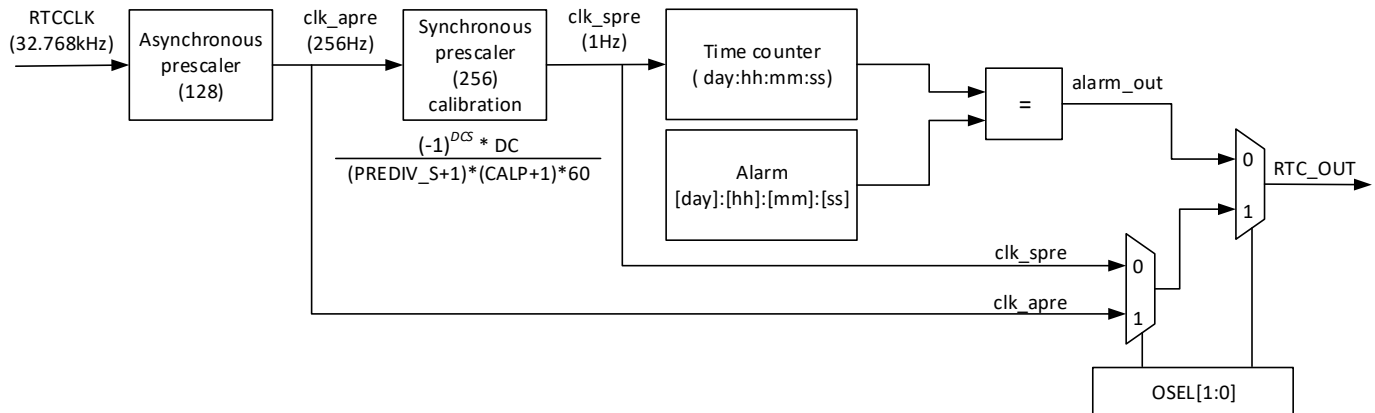


Fig 11-2 RTC prescale diagram

11.1.4 RTC Clock Select Diagram

The RTC clock select diagram is shown in Fig 11-3.

- When system boots, the default RTC clock source (RTCCLK) is from SDM32K with 0x4800_0004[13:12] = 0 and 0x4800_0004[8] = 0.
- When setting 0x4800_0004[13:12] = 10/11 and 0x4800_0004[8] = 0, it means that EXT32K is selected as RTC clock source.
- When setting 0x4800_0004[8] = 1 but no care for 0x4800_0004[13:12], the configure divider data for XTAL32K output is from XTAL40MHz.
- When setting 0x4800_0004[13:12] = 01 and 0x4800_0004[8] = 0, the RTC clock source shall be 131kHz. This channel can be selected when testing for the precision of 131kHz.
- When switching RTCCLK between XTAL32K and SDM32K/EXT32K with setting 0x4800_0004[8], you must keep the clock exit in the two channels. When switching has done, the other channel unused can be power off if needed.

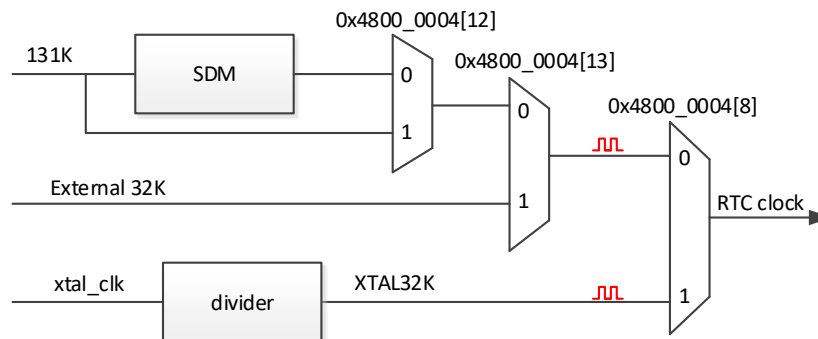


Fig 11-3 RTC clock select diagram

11.2 Functional Description

11.2.1 Clock and Prescaler

A programmable prescaler stage generates a 1Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers.

- A 9-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 9-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: It is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

Default, the asynchronous prescaler division factor is set to 128, and the synchronous division factor is set to 256, to obtain an internal clock frequency of 1Hz (clk_spre) with 32.768kHz as RTCCLK.

f_{clk_apre} is given by the following formula:

$$f_{clk_apre} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

f_{clk_spre} is given by the following formula:

$$f_{clk_spre} = \frac{f_{clk_apre}}{PREDIV_S + 1}$$

11.2.2 32K Auto-trigger Calibration Circuit

RTCCLK is supported by SDM module, and the clock source is calibrated with xtal_clk.

When system is in high speed mode, the SDM module can always be calibrated with xtal_clk. When system is in low power mode, RTC can wake up PMC to open xtal_clk by generating xtal_req_o signal. The xtal_req_32k (xtal_req_o) is also given to SDM module to inform it that the calibration begins. When xtal_clk is stable, the SDM module receives a xtal_valid signal from PMC and begins to do calibration. When calibration has finished, SDM gives a cal_done_100k signal to inform RTC to pull down the xtal_req_32k (xtal_req_o) signal and gives xtal_valid_ack to PMC. PMC then enters low power mode when receiving the xtal_valid_ack signal (See Fig 11-4).

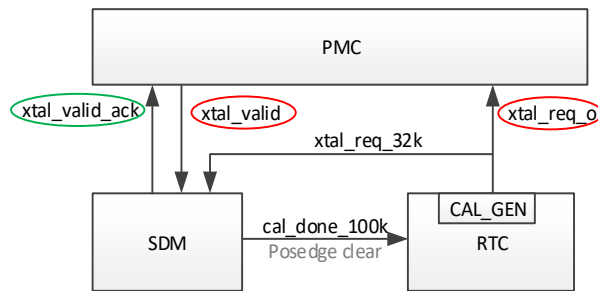


Fig 11-4 Calibration block diagram

The RTC unit provides auto calibration function. It can be selected with ACAL_SEL[1:0] and a programmable threshold with ACAL_THES[5:0] in RTC_CLKACALR register to generate a xtal_req_32k request, as Fig 11-5 shows.

- ACAL_SEL[1:0] = 00, the SDM32K auto calibration function is disabled.
- ACAL_SEL[1:0] = 01 and the ACAL_THES[5:0] is set, the ACAL_CNT[5:0] updates once per minute.
- ACAL_SEL[1:0] = 10 and the ACAL_THES[5:0] is set, the ACAL_CNT[5:0] updates once per hour.
- ACAL_SEL[1:0] = 11 and the ACAL_THES[5:0] is set, the ACAL_CNT[5:0] updates once per day.

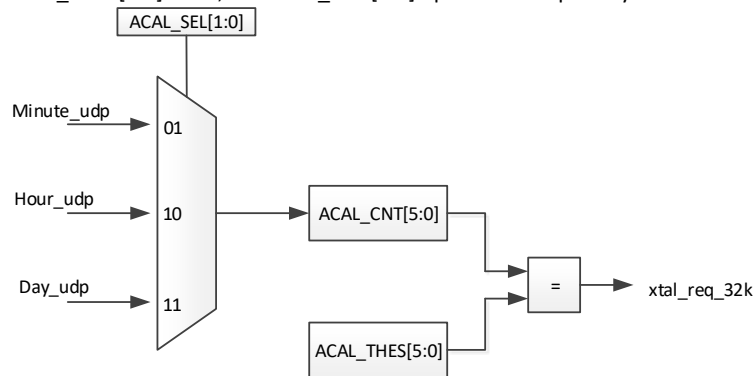


Fig 11-5 xtal_req_32k block diagram

When the ACAL_CNT[5:0] value equals to the ACAL_THES[5:0], the xtal_req_32k is generated and the SDM32K calibration circuit works automatically.

11.2.3 Programmable Alarm

The RTC unit provides one programmable alarm.

The programmable alarm function is enabled through the ALME bit in the RTC_CR register. The ALMF is set to 1 if the calendar seconds, minutes, hours or days match the values programmed in the alarm registers RTC_ALMR1L and RTL_ALMR1H. Each calendar field can be independently selected through the MSKx bits. The alarm interrupt is enabled through the ALMIE bit in the RTC_CR register.

Alarm (if enabled by the OSEL[1:0] bits in RTC_CR) can be routed to the RTC_OUT output. The alarm output is a pulse which width is 1/RTCCLK.

11.2.4 Write Protection

After RTC domain reset, all the RTC registers are write-protected. Writing to the RTC registers is enabled by writing a key into the RTC_WPR register.

The following steps are required to unlock the write protection on all the RTC registers except for ALMF in RTC_ISR.

- (1) Write '0xCA' into the RTC_WPR register.
- (2) Write '0x53' into the RTC_WPR register.

Writing a wrong key reactivates the write protection. The protection mechanism isn't affected by system reset.

11.2.5 Digital Calibration

The digital calibration can be used to compensate RTCCLK by adding (positive calibration) or masking (negative calibration) clock cycles at the output of the asynchronous prescaler (clk_apre).

Positive and negative calibrations are selected by setting the DCS bit to '0' and '1' in RTC_CALIBR register, respectively.

- When positive calibration is enabled (DCS = '0'), DC clk_apre cycle is added every (CALP+1) minutes. This causes the calendar to be updated sooner, thereby adjusting the effective RTC frequency to be a bit higher.
- When negative calibration is enabled (DCS = '1'), DC clk_apre cycle is removed every (CALP +1) minutes. This causes the calendar to be updated later, thereby adjusting the effective RTC frequency to be a bit lower.

DC and CALP can be configured through the RTC_CALIBR register. DC must be less than PREDIV_S in the RTC_PRER register.

The calibration parameter can be configured on-the-fly. Calibrating resolution is determined by the frequency of clk_apre and the calibration period. The example is shown in Table 11-1.

Table 11-1 Example of calibrating resolution

CALP	clk_apre (128Hz)	clk_apre (256Hz)	clk_apre (512Hz)
1 min	130.2ppm	65.1ppm	32.55ppm
2 min	65.1ppm	32.55ppm	16.27ppm
4 min	32.55ppm	16.27ppm	8.14ppm
8 min	16.27ppm	8.14ppm	4.07ppm

Re-calibration on-the-fly:

The calibration register (RTC_CALIBR) can be updated on the fly while INITF=0, by using the following steps:

- (1) Poll the RECALPF (re-calibration pending flag).
- (2) If RECALPF is set to 0, write a new value to RTC_CALIBR if necessary, RECALPF is then automatically set to 1.
- (3) Within three clk_apre cycles after the write operation to RTC_CALIBR, the new calibration settings take effect.

11.2.6 Day Threshold Program

The RTC provides day interrupt for users. To use day interrupt, DAY_THRES[8:0] must be programmed first in [RTC_CR](#) and then set the DOVTHIE bit in RTC_CR to enable day over threshold interrupt.

11.3 Registers

The physical base address of RTC is 0x4800 4000.

Table 11-2 Register table of RTC

Name	Address Offset	Access	Description
RTC_TR	0x00	R/W	RTC time register
RTC_CR	0x04	R/W	RTC control register
RTC_ISR	0x08	R/W	RTC initialization and status register
RTC_PRER	0x0C	R/W	RTC prescaler register
RTC_CALIBR	0x10	R/W	RTC calibration register
RTC_ALMR1L	0x14	R/W	RTC alarm1 register low
RTC_ALMR1H	0x18	R/W	RTC alarm1 register high
RTC_WPR	0x1C	R/W	RTC write protection register
RTC_CLKACALR	0x20	R/W	RTC 32K auto calibration register

11.3.1 RTC Time Register (RTC_TR)

- **Name:** RTC time register
- **Size:** 32 bits
- **Address offset:** 0x00
- **Reset value:** 0x0000 0000

This register is the calendar time shadow register. This register is write protected, and must be written in initialization mode only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAY[8:0]									PM	HT[1:0]		HU[3:0]			
R/W									R/W	R/W		R/W			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	MNT[2:0]			MNU[3:0]				RSVD	ST[2:0]			SU[3:0]			
	R/W			R/W					R/W			R/W			

Bit	Name	Access	Reset	Description
31:23	DAY[8:0]	R/W	0	Day in binary format
22	PM	R/W	0	<ul style="list-style-type: none"> ● AM/PM notation ● 0: AM or 24-hour format ● 1: PM
21:20	HT[1:0]	R/W	0	Hour tens in BCD format
19:16	HU[3:0]	R/W	0	Hour units in BCD format
15	RSVD	N/A	-	Reserved
14:12	MNT[2:0]	R/W	0	Minute tens in BCD format
11:8	MNU[3:0]	R/W	0	Minute units in BCD format
7	RSVD	N/A	-	Reserved
6:4	ST[2:0]	R/W	0	Second tens in BCD format
3:0	SU[3:0]	R/W	0	Second units in BCD format

11.3.2 RTC Control Register (RTC_CR)

- **Name:** RTC control register

- **Size:** 32 bits
- **Address offset:** 0x04
- **Reset value:** 0xFF80 0000

This register is write protected. Bit[7] (FMT) of this register can be written in initialization mode only when INITF = 1. ADD1H and SUB1H changes are effective in 2~3 seconds. Don't write this register continuously without any delay when RTC is in free run mode. Software can use the RSF bit in RTC_ISR register to handle the delay.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAY_THRES[8:0]									RSVD						DOVTHIE
R/W															R/W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD			ALMIE	RSVD			ALME	FMT	OSEL[1:0]		RSVD	BYP SHAD	BKP	SUB1H	ADD1H
			R/W				R/W	R/W	R/W	R/W		R/W	R/W	W	W

Bit	Name	Access	Reset	Description
31:23	DAY_THRES[8:0]	R/W	0x1FF	Day threshold in binary format
22:17	RSVD	N/A	-	Reserved
16	DOVTHIE	R/W	0	Day over threshold interrupt enable <ul style="list-style-type: none"> ● 0: Day over threshold interrupt is disabled ● 1: Day over threshold interrupt is enabled
15:13	RSVD	N/A	-	Reserved
12	ALMIE	R/W	0	Alarm interrupt enable <ul style="list-style-type: none"> ● 0: Alarm interrupt is disabled ● 1: Alarm interrupt is enabled
11:9	RSVD	N/A	-	Reserved
8	ALME	R/W	0	Alarm enable <ul style="list-style-type: none"> ● 0: Alarm is disabled ● 1: Alarm is enabled
7	FMT	R/W	0	Hour format <ul style="list-style-type: none"> ● 0: 24 hour/day format ● 1: AM/PM hour format
6:5	OSEL[1:0]	R/W	0	Output selection These bits are used to select the flag to be routed to RTC_OUT output <ul style="list-style-type: none"> ● 00: Output is disabled (logic 0) ● 01: Alarm output is enabled ● 10: Clock output is clk_spre (default: 1Hz) ● 11: Clock output is clk_apre (default: 512Hz)
4	RSVD	N/A	-	Reserved
3	BYP SHAD	R/W	0	Bypass the shadow registers <ul style="list-style-type: none"> ● 0: Calendar values (when reading from RTC_TR) are taken from the shadow registers, which are updated once every two RTCCLK cycles. ● 1: Calendar values (when reading from RTC_TR) are taken directly from the calendar counters.
2	BKP	R/W	0	Backup This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.
1	SUB1H	W1	0	Subtract 1 hour (winter time change) When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour isn't 0. Setting this bit has no effect when current hour is 0. <ul style="list-style-type: none"> ● 0: No effect ● 1: Subtracts 1 hour to the current time. This can be used for winter time change. This bit is always read as 0.
0	ADD1H	W1	0	Add 1 hour (summer time change) When this bit is set outside initialization mode, 1 hour is added to the calendar time. <ul style="list-style-type: none"> ● 0: No effect ● 1: Adds 1 hour to the current time. This can be used for summer time change. This bit is always read as 0.

11.3.3 RTC Initialization and Status Register (RTC_ISR)

- **Name:** RTC initialization and status register
- **Size:** 32 bits
- **Address offset:** 0x08
- **Reset value:** 0x0000 0000

The ALMF bit can be written without unlocking the write protection. Two APB clock cycles after programming it to 1, this bit is cleaned.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															RECALPF
															R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOVTHF	RSVD						ALMF	INIT	INITF	RSF	INITS	RSVD			ALMWF
R/W1C							R/W1C	R/W	R	R/W1C	R				R

Bit	Name	Access	Reset	Description
31:17	RSVD	N/A	-	Reserved
16	RECALPF	R	0	Recalibration pending Flag The RECALPF status flag is automatically set to '1' when software writes to the RTC_CALIBR register, indicating that the RTC_CALIBR register is blocked. When the new calibration settings are taken into account, this bit returns to '0'. Refer to Re-calibration on-the-fly.
15	DOVTHF	R/W1C	0	Day over threshold flag This flag is set by hardware when the Day[8:0] in RTC_TR over the DAY_THRES[8:0] set in RTC_CR register.
14:9	RSVD	N/A	-	Reserved
8	ALMF	R/W1C	0	Alarm flag This flag is set by hardware when the time register (RTC_TR) match the alarm registers (RTC_ALMR1L and RTC_ALMR1H).
7	INIT	R/W	0	Initialization mode <ul style="list-style-type: none"> ● 0: Free running mode ● 1: Initialization mode used to program time and date register (RTC_TR), and prescaler register (RTC_PRER). Counters stop and start counting from the new value when INIT is set.
6	INITF	R	0	Initialization flag When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated. <ul style="list-style-type: none"> ● 0: Calendar registers update isn't allowed ● 1: Calendar registers update is allowed
5	RSF	R/W1C	0	Registers synchronization flag This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_TR). This bit is cleared by hardware in initialization mode or when in bypass shadow register mode (BYP SHAD=1). This bit can also be cleared by software. It is cleared either by software or by hardware in initialization mode. <ul style="list-style-type: none"> ● 0: Calendar shadow registers hasn't yet synchronized ● 1: Calendar shadow registers has synchronized
4	INITS	R	0	This bit is set by hardware when the calendar day field is different from 0 (RTC domain reset state). <ul style="list-style-type: none"> ● 0: Calendar hasn't been initialized ● 1: Calendar has been initialized
3:1	RSVD	N/A	-	Reserved
0	ALMWF	R	0	Alarm write flag This bit is set by hardware when alarm values can be changed, after the ALME bit has been set to 0 in RTC_CR register. It is cleared by hardware when ALME bit has been set to 1 in RTC_CR register. <ul style="list-style-type: none"> ● 0: Alarm update isn't allowed ● 1: Alarm update is allowed

11.3.4 RTC Prescaler Register (RTC_PRER)

- **Name:** RTC prescaler register
- **Size:** 32 bits
- **Address offset:** 0x0C
- **Reset value:** 0x007F 00FF

This register is write protected, and must be written in initialization mode only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								PREDIV_A							
								R/W							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								PREDIV_S							
								R/W							

Bit	Name	Access	Reset	Description
31:25	RSVD	N/A	-	Reserved
24:16	PREDIV_A	R/W	0x7F	Asynchronous prescaler factor This is the asynchronous division factor: $f_{clk_apre} = RTCCLK / (PREDIV_A + 1)$
15:9	RSVD	N/A	-	Reserved
8:0	PREDIV_S	R/W	0xFF	Synchronous prescaler factor This is the synchronous division factor: $f_{clk_spre} = f_{clk_apre} / (PREDIV_S + 1)$

11.3.5 RTC Calibration Register (RTC_CALIBR)

- **Name:** RTC calibration register
- **Size:** 32 bits
- **Address offset:** 0x10
- **Reset value:** 0x0000 0000

This register is write protected, and can be dynamically configured when RTC is running.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD													CALP[2:0]		
													R/W		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCE	DCS	RSVD							DC						
R/W	R/W								R/W						

Bit	Name	Access	Reset	Description
31:19	RSVD	N/A	-	Reserved
18:16	CALP[2:0]	R/W	0	Calibration period Compensate $\frac{(-1)^{DCS} \cdot DC}{PREDIV_S + 1}$ seconds every (CALP+1) minutes
15	DCE	R/W	0	Digital calibration enable <ul style="list-style-type: none"> ● 0: Digital calibration is disabled ● 1: Digital calibration is enabled
14	DCS	R/W	0	Digital calibration sign <ul style="list-style-type: none"> ● 0: Positive calibration: time update frequency is increased ● 1: Negative calibration: time update frequency is decreased
13:7	RSVD	N/A	-	Reserved
6:0	DC	R/W	0	Digital calibration

11.3.6 RTC Alarm 1 Register Low (RTC_ALMR1L)

- **Name:** RTC alarm 1 register low
- **Size:** 32 bits
- **Address offset:** 0x14
- **Reset value:** 0x0000 0000

This register is write protected, and can be written only when ALMWF is set to 1 in RTC_ISR register, or in initialization mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								MSK2	PM	HT[1:0]	HU[3:0]				
								R/W	R/W	R/W		R/W			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK1	MNT[2:0]			MNU[3:0]				MSK0	ST[2:0]			SU[3:0]			
R/W	R/W			R/W				R/W	R/W			R/W			

Bit	Name	Access	Reset	Description
31:24	RSVD	N/A	-	Reserved
23	MSK2	R/W	0	Alarm hour mask <ul style="list-style-type: none"> ● 0: Alarm set if the hour matches ● 1: Hours don't care in alarm comparison
22	PM	R/W	0	AM/PM notation <ul style="list-style-type: none"> ● 0: AM or 24-hour format ● 1: PM
21:20	HT[1:0]	R/W	0	Hour tens in BCD format.
19:16	HU[3:0]	R/W	0	Hour units in BCD format.
15	MSK1	R/W	0	Alarm minute mask <ul style="list-style-type: none"> ● 0: Alarm set if the minute matches ● 1: Minutes don't care in alarm comparison
14:12	MNT[2:0]	R/W	0	Minute tens in BCD format.
11:8	MNU[3:0]	R/W	0	Minute units in BCD format.
7	MSK0	R/W	0	Alarm second mask <ul style="list-style-type: none"> ● 0: Alarm set if the second matches ● 1: Seconds don't care in alarm comparison
6:4	ST[2:0]	R/W	0	Second tens in BCD format.
3:0	SU[3:0]	R/W	0	Second units in BCD format.

11.3.7 RTC Alarm 1 Register High (RTC_ALMR1H)

- **Name:** RTC alarm 1 register high
- **Size:** 32 bits
- **Address offset:** 0x18
- **Reset value:** 0x0000 0000

This register is write protected, and can be written only when ALMWF is set to 1 in RTC_ISR register, or in initialization mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							MSK3	DAY[8:0]							
							R/W	R/W							

Bit	Name	Access	Reset	Description
31:10	RSVD	N/A	-	Reserved

9	MSK3	R/W	0	Alarm day mask <ul style="list-style-type: none"> 0: Alarm set if the day matches 1: Days don't care in alarm comparison
8:0	DAY[8:0]	R/W	0	Day in binary format

11.3.8 RTC Write Protection Register (RTC_WPR)

- **Name:** RTC write protection register
- **Size:** 32 bits
- **Address offset:** 0x1C
- **Reset value:** 0x0000 0000

31	30	29	...	10	9	8	7	6	5	4	3	2	1	0
RSVD							KEY[7:0]							
							R/W							

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7:0	KEY[7:0]	R/W	0	Write protection key This byte is written by software. Refer to RTC register write protection for a description of how to unlock RTC register write protection.

11.3.9 RTC 32K Auto-calibration Register (RTC_CLKACALR)

- **Name:** RTC 32K auto-calibration register
- **Size:** 32 bits
- **Address offset:** 0x20
- **Reset value:** 0x0000 00FC

This register is write protected, and must be written in initialization mode only. The ACAL_CNT[5:0] counter is cleared after exiting the initialization mode and waiting until RSF=1 in RTC_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD										ACAL_CNT[5:0]					
										R					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								ACAL_THES[5:0]						ACAL_SEL[1:0]	
								R/W						R/W	

Bit	Name	Access	Reset	Description
31:22	RSVD	N/A	-	Reserved
21:16	ACAL_CNT[5:0]	R	0	Auto Calibration Counter The counter update depends on ACAL_SEL[1:0]. <ul style="list-style-type: none"> ACAL_SEL[1:0] = 00: Counter is disabled ACAL_SEL[1:0] = 01: Counter updates once per minute ACAL_SEL[1:0] = 10: Counter updates once per hour ACAL_SEL[1:0] = 11: Counter updates once per day
15:8	RSVD	N/A	-	Reserved
7:2	ACAL_THRES[5:0]	R/W	0x3f	Auto Calibration Threshold Note: The unit depends on ACAL_SEL[1:0].
1:0	ACAL_SEL[1:0]	R/W	0	32K Auto Calibration select <ul style="list-style-type: none"> 00: 32K auto calibration is disabled and the auto calibration counter is cleared. 01: 32K auto calibration once every ACAL_THRES[5:0] minutes 10: 32K auto calibration once every ACAL_THRES[5:0] hours 11: 32K auto calibration once every ACAL_THRES[5:0] days

11.4 Operation Flow

11.4.1 Initialize the Calendar

The initialization calendar configure flow is shown in Table 11-3.

Table 11-3 Initialization calendar configure flow

Step	What to do	How to do	Comments
1	Disable the RTC registers write protection	Write '0xCA' and then '0x53' into the RTC_WPR register	RTC registers can be modified
2	Enter Initialization mode	Set INIT bit to '1' in RTC_ISR register	The calendar counter is stopped to allow update
3	Wait for the confirmation of Initialization mode (clock synchronization)	Poll INITF bit of in RTC_ISR until it is set	It takes approximately 2 RTCCLK clock cycles for medium density devices
4	Program the prescaler register if needed	Configure RTC_PRER register	By default, the RTC_PRER prescalers register is initialized to provide 1Hz to the Calendar unit when RTCCLK = 32.768kHz
5	Load time value in the shadow registers	Set RTC_TR	-
6	Configure the time format (12h or 24h)	Set FMT bit in RTC_CR register	FMT = 0: 24 hour/day format FMT = 1: AM/PM hour format
7	Exit Initialization mode	Clear the INIT bit in RTC_ISR register	The current calendar counter is automatically loaded and the counting restarts after 4 RTCCLK cycles
8	Enable the RTC registers Write protection	Write '0xFF' into the RTC_WPR register	RTC registers can no longer be modified

11.4.2 Configure Alarm

The alarm configure flow is shown in Table 11-4.

Table 11-4 Alarm configure flow

Step	What to do	How to do	Comments
1	Disable the RTC registers write protection	Write '0xCA' and then '0x53' into the RTC_WPR register	RTC registers can be modified
2	Disable alarm	Clear the ALME bit in RTC_CR register	-
3	Check that the RTC_ALRMA register can be accessed	Poll the ALMWF bit until it is set in RTC_ISR register.	It takes approximately two RTCCLK clock cycles (clock synchronization).
4	Configure the alarm	Configure RTC_ALMR1L and RTC_ALMR1H register	The alarm hour format must be the same as the RTC Calendar
5	Re-enable alarm	Set the ALME bit in RTC_CR register	-
6	Enable the RTC registers write protection	Write "0xFF" into RTC_WPR register	RTC registers can no longer be modified

Note:

- The alarm behaviour can be configured using the MSKx bits ($x = \{0, 1, 2, 3\}$) of the RTC_ALMR1L and RTC_ALMR1H registers.
- Alarm can also be configured in the initialization mode.

11.4.3 Configure Calibration

The calibration configure flow is shown in Table 11-5.

Table 11-5 Calibration configure flow

Step	What to do	How to do	Comments
------	------------	-----------	----------

1	Disable the RTC registers write protection	Write '0xCA' and then '0x53' into the RTC_WPR register	RTC registers can be modified
2	Check that the RTC_CALIBR register can be updated	Poll the RECALPF bit until it is 0 in RTC_ISR register	-
3	Configure the calibration parameter	Configure RTC_CALIBR register	-
4	Enable the RTC registers Write protection	Write '0xFF' into the RTC_WPR register	RTC registers can no longer be modified

Note:

- Calibration can also be configured in the initialization mode.
- The calibration parameter isn't used immediately. Hardware automatically reloads the new calibration parameter at the end of calibration period.

11.4.4 Daylight Saving Time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

It is recommended not to change the hour during the calendar hour increment, because hardware can handle this condition. If this condition occurs, hardware delays the SUB1H or ADD1H operation for one second.

In the U.S., 2:00 am is originally chosen as the changeover time because it is practical and minimized disruption. Most people are at home and this is the time when the fewest trains are running. It is late enough to minimally affect bars and restaurants, and it prevents the day from switching to yesterday, which is confusing. It is early enough that the entire continental U.S. switches by daybreak, and the changeover occurs before most early shift workers and early churchgoers are affected. (<http://www.webexhibits.org/daylightsaving/b.html>)

12 Watchdog Timer (WDT)

12.1 Introduction

The Watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates a MCU reset or a WDG interrupt on the expiry of a programmed time period, unless the program refreshes the contents of the watchdog timer counter before it's overflow.

When a MCU reset happens, the watchdog timer counter is always cleared and the watchdog is always disabled.

12.2 Features

- Reset mode: The watchdog circuit generates a MCU reset on the expiry of a programmed time period, unless the program refreshes the watchdog.
- Interrupt mode: The watchdog circuit generates a WDG interrupt on the expiry of a programmed time period, unless the program refreshes the watchdog.

Clock source: The watchdog is supported by 32.768kHz.

- Timeout formula:

$$Timeout = \frac{1}{\frac{32.768kHz}{DivFactor + 1}} * Count$$

12.3 Registers

The base address of WDT register is:

- KM4: 0x4000_2800
- KM0: 0x4800_2800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDT_TO	WDT_MODE	RSVD	COUNTID				WDT_CLEAR	RSVD							WDT_EN_BIT
R/W1C	R/W		R/W				W								R/W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIT_VNDR_DIVFACTOR															
R/W															

Bit	Name	Access	Reset	Description
31	WDT_TO	R/W1C	0	Watchdog timer timeout. 1 cycle pulse.
30	WDT_MODE	R/W	0	<ul style="list-style-type: none"> 0: Interrupt CPU when WDT timer counter is overflow. 1: Reset system when WDT timer counter is overflow.
29	RSVD	N/A	0	Reserved
28:25	COUNTID	R/W	0	<ul style="list-style-type: none"> 0: 0x001 1: 0x003 2: 0x007 3: 0x00F 4: 0x01F 5: 0x03F 6: 0x07F 7: 0x0FF 8: 0x1FF 9: 0x3FF 10: 0x7FF 11~15: 0xFFF

				Watchdog Count= (0x00000001 << (COUNTID + 1)) - 1
24	WDT_CLEAR	W	0	Refresh watchdog
23:17	RSVD	N/A	0	Reserved
16	WDT_EN_BIT	R/W	0	<ul style="list-style-type: none"> ● 1: Enable watchdog timer ● 0: Disable watchdog timer
15:0	BIT_VNDR_DIVFACTOR	R/W	1	Dividing factor Watchdog timer counts with 32.768kHz/(DivFactor + 1), the minimum dividing factor is 1. The formula of dividing factor computation is: $\text{DivFactor} = (\text{u16}) (\text{Timeout} * 100) / (\text{Count} * 3)$

13 Inter-integrated Circuit (I²C) Interface

13.1 Product Introduction

This chapter describes the Ameba-D I²C Interface Peripheral, referred to as I²C.

The design of Ameba-D I²C aims on sensor hub application in low-power or battery-powered productions. Essential features of I²C bus protocol should be provided for acquiring or controlling external sensor data. To reduce system power consumption, advanced application scheme in Ameba-D I²C are also available for further low power system state.

Ameba-D I²C has the following features:

- Two-wire I²C serial interface – consists of a serial data line (SDA) and a serial clock (SCL)
- Support one I²C port
- Two Speed mode:
 - Standard (up to 100Kbps)
 - fast (up to 400Kbps)
- Master or Slave I²C operation
- 7- or 10-bit addressing
- Transmit and receive buffers with depth of 16
- Tx and Rx DMA support
- Multi-master ability including bus arbitration scheme
- Slave mode address match wakeup for power save (up to 100kbps)
- Clock stretch in master/slave mode
- 7- or 10-bit combined format transfers
- General Call
- Component parameters for configurable software driver support (programmable SDA hold time, slave address, etc.)
- Filter to eliminate the glitches on signal of SDA and SCL. Programmable digital noise Filter
- Status flags (Bus busy flag, activity flag, FIFO status flag, etc.) and Error flags (arbitration lost, acknowledge failure, etc.)

13.2 Functional Description

This chapter describes the functional behavior of Ameba-D I²C in more details.

13.2.1 Overview

The I²C bus is a two-wire serial interface, consisting of a serial data line (SDA) and a serial clock (SCL). These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a “transmitter” or “receiver,” depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. A master is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

Note: The Ameba-D I²C must only be programmed to operate in either master or slave mode only. Operating as master and slave simultaneously is not supported.

Any I²C device can be attached to an I²C-bus and every device can talk with any master, passing information back and forth. There needs to be at least one master (such as a microcontroller or DSP) on the bus but there can be multiple masters, which require them to arbitrate for ownership. Multiple masters and arbitration are explained later in this chapter.

There are three clock domains, APB clock, Core clock and SCL clock.

- Ameba-D I²C uses bus clock as APB clock, which has frequency of 10M.
- Core clock is consisted of bus clock (10M) and OSC clock (2M), which is selected by a mux. Ameba-D I²C uses bus clock as core clock expect for low power mode.
- SCL clock is from the I²C bus, which is 100K/s in Standard Mode, 400K/s in Fast Mode and 3.4M/s in High Speed Mode (not supported in Ameba-D).

The following defines the file names and functions of the blocks in Fig 13-1.

- APB Interface: It takes the APB interface signals and translates them into a common generic interface that allows the register file to be bus protocol-agnostic.
- DMA Interface: I²C has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA.
- DMA Parse: It controls the operation mode of DMA.
- TRX FIFO: It holds the Rx FIFO and Tx FIFO register banks and controllers, along with their status levels.
- Shift Register: It has two functions, Rx shift register and Tx shift register. Rx shift register takes data into the design and extracts it in byte format. Tx shift register presents data supplied by CPU for transfer on the I²C bus.
- Master: It generates the protocol for the master transfers.
- Slave: It follows the protocol for a slave and monitors bus for address match.
- Control Register: It contains configuration registers and is the interface with software.
- Synchronous Module: It transfers signals from APB clock domain to ic_clk domain.
- CLKRST Generator: It adds clock gating to I²C to reduce power consumption.
- Bus Monitor: It calculates SCL cycles and detects the bus state.
- Detector: It detects the events in the bus; for example, positive/negative edge of SDA/SCL, ACK/NACK on SDA and etc.
- Output Control: It controls the outputs of SDA line and SCL line in the bus.
- Rx Filter: It is a Rx filter to avoid glitches when I²C receives data from the bus.

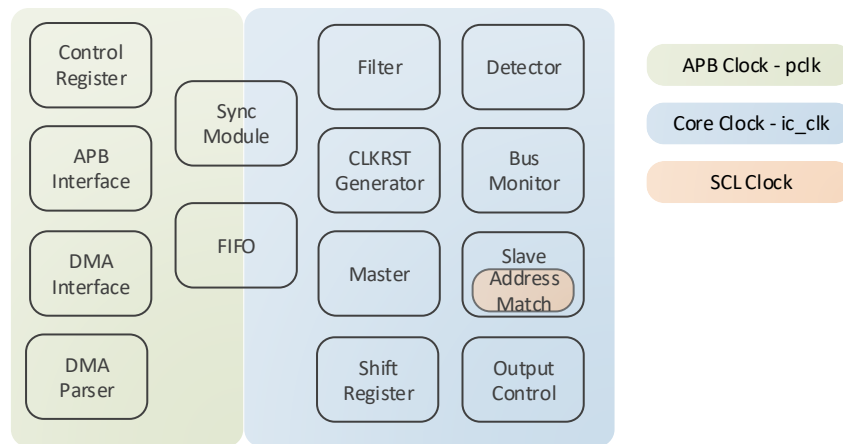


Fig 13-1 Block diagram of I²C

Note: The ic_clk frequency must be greater than or equal to the pclk frequency. This restriction occurs because the clock domain-crossing scheme within the I²C does not support pclk faster than ic_clk.

13.2.2 I²C Terminology

The following terms are used throughout this manual and are defined as follows:

13.2.2.1 I²C Bus Terms

The following terms relate to how the role of the I²C device and how it interacts with other I²C devices on the bus.

- **Transmitter** – the device that sends data to the bus. A transmitter can either be a device that initiates the data transmission to the bus (a *master-transmitter*) or responds to a request from the master to send data to the bus (a *slave-transmitter*).
- **Receiver** – the device that receives data from the bus. A receiver can either be a device that receives data on its own request (a *master-receiver*) or in response to a request from the master (a *slave-receiver*).
- **Master** – the component that initializes a transfer (START command), generates the clock (SCL) signal and terminates the transfer (STOP command). A master can be either a transmitter or a receiver.
- **Slave** – the device addressed by the master. A slave can be either receiver or transmitter. These concepts are illustrated in Fig 13-2.
- **Multi-master** – the ability for more than one master to co-exist on the bus at the same time without collision or data loss.
- **Arbitration** – the predefined procedure that authorizes only one master at a time to take control of the bus.

- **Synchronization** – the predefined procedure that synchronizes the clock signals provided by two or more masters.
- **SDA** – data signal line (Serial Data)
- **SCL** – clock signal line (Serial Clock)

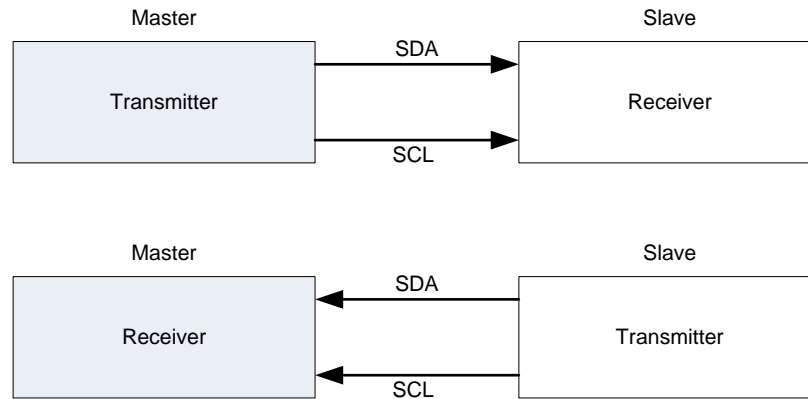


Fig 13-2 Master/Slave and Transmitter/Receiver relationships

13.2.2.2 Bus Transfer Terms

The following terms are specific to data transfers that occur to/from the I²C bus.

- **START (RESTART)** – data transfer begins with a START or RESTART condition. The level of the SDA data line changes from high to low, while the SCL clock line remains high. When this occurs, the bus becomes busy.
Note: START and RESTART conditions are functionally identical.
- **STOP** – data transfer is terminated by a STOP condition. This occurs when the level on the SDA data line passes from the low state to the high state, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free or idle once again. The bus stays busy if a RESTART is generated instead of a STOP condition.

13.2.3 I²C Behavior

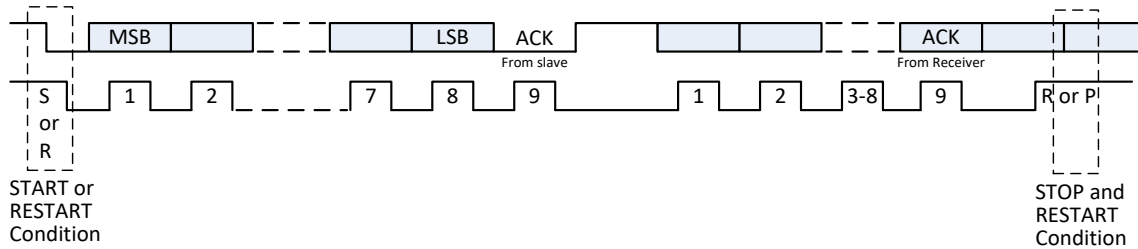
The I²C can be controlled via software to be either:

- An I²C master only, communicating with other I²C slaves; OR
- An I²C slave only, communicating with one more I²C masters.

The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. The acknowledgement of data is sent by the device that is receiving data, which can be either a master or a slave. As mentioned previously, the I²C protocol also allows multiple masters to reside on the I²C bus and uses an arbitration procedure to determine bus ownership.

Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that is then followed by the slave's address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge (ACK) pulse after the address.

If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver gets one byte of data. This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with the ACK pulse. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition. This behavior is illustrated in Fig 9-9.


Fig 13-3 Data transfer on the I²C bus

The I²C is a synchronous serial interface. The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP, START, and RESTART conditions. The output drivers are open-drain or open-collector to perform wire-AND functions on the bus. The maximum number of devices on the bus is limited by only the maximum capacitance specification of 400pF. Data is transmitted in byte packages.

13.2.3.1 START and STOP Generation

When operating as an I²C master, putting data into the transmit FIFO causes the Ameba-D I²C to generate a START condition on the I²C bus. Writing a 1 to IC_DATA_CMD[9] causes the hardware to generate a STOP condition on the I²C bus; a STOP condition is not issued if this bit is not set, even if the transmit FIFO is empty. Writing a 1 to IC_DATA_CMD[10] causes the hardware to hold bus after the current data is transmitted and generate a RESTART condition when the next data in FIFO is ready to be transmitted on bus.

When operating as a slave, the Ameba-D I²C does not generate START and STOP conditions, as per the protocol. However, if a read request is made to the Ameba-D I²C, it holds the SCL line low until read data has been supplied to it. This stalls the I²C bus until read data is provided to the slave Ameba-D I²C, or the Ameba-D I²C slave is disabled by writing a 0 to bit0 of the REG_IC_ENABLE.

13.2.3.2 Combined Formats

The Ameba-D I²C supports mixed read and write combined format transactions in both 7-bit and 10-bit addressing modes.

The I²C does not support mixed address and mixed address format—that is, a 7-bit address transaction followed by a 10-bit address transaction or vice versa—combined format transactions.

To initiate combined format transfers, IC_CON.IC_RESTART_EN should be set to 1. With this value set and operating as a master, when the I²C completes an I²C transfer, it checks the transmit FIFO and executes the next transfer. If the direction of this transfer differs from the previous transfer, the combined format is used to issue the transfer. If the transmit FIFO is empty when the current I²C transfer completes, IC_DATA_CMD[9] is checked.

- If set to 1, a STOP bit is issued.
- If set to 0, the SCL is hold low until the next command is written to the transmit FIFO.

13.2.4 I²C Protocols

The I²C has the protocols discussed in this section.

13.2.4.1 START and STOP Conditions

When the bus is idle, both the SCL and SDA signals are pulled high through external pull-up resistors on the bus. When the master wants to start a transmission on the bus, the master issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is 1. When the master wants to terminate the transmission, the master issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1. Fig 9-10 shows the timing of the START and STOP conditions. When data is being transmitted on the bus, the SDA line must be stable when SCL is 1.

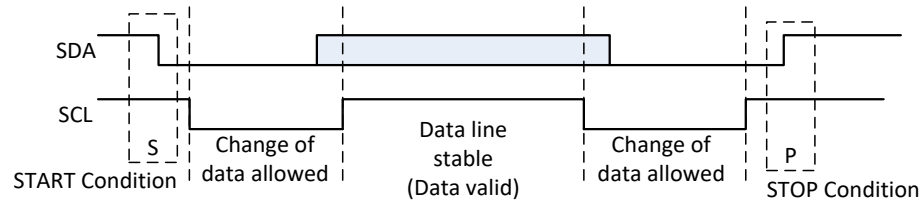


Fig 13-4 START and STOP conditions

13.2.4.2 Addressing Slave Protocol

There are two address formats: the 7-bit address format and the 10-bit address format.

13.2.4.2.1 7-bit Address Format

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the slave address and the LSB bit (bit 0) is the R/W bit as shown in Fig 9-11. When bit 0 (R/W) is set to 0, the master writes to the slave. When bit 0 (R/W) is set to 1, the master reads from the slave.

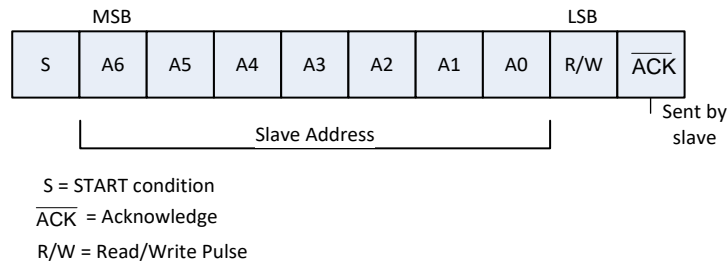


Fig 13-5 7-bit address format

13.2.4.2.2 10-bit Address Format

During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the slaves address bits 9:8, and the LSB bit (bit 0) is the R/W bit. The second byte transferred sets bits 7:0 of the slave address. Fig 9-12 shows the 10-bit address format, and Table 13-1 defines the special purpose and reserved first byte addresses.

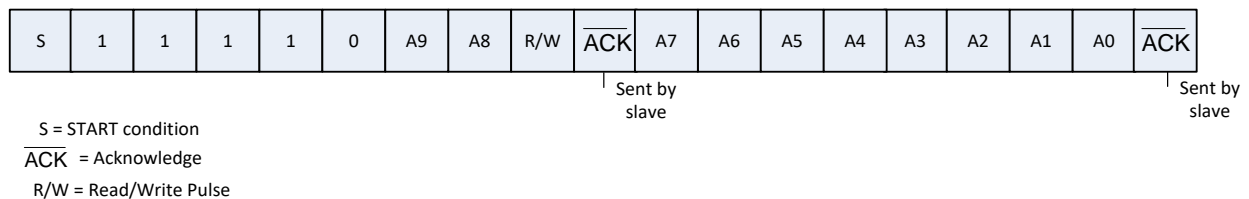


Fig 13-6 10-bit address format

Table 13-1 I²C definition of bits in the first byte

Slave Address	R/W Bit	Description
0000 000	0	General Call Address. I ² C places the data in the receive buffer and issues a General Call interrupt.
0000 000	1	START byte.
0000 001	X	CBUS address. I ² C ignores these accesses.
0000 010	X	Reserved
0000 011	X	Reserved

0000 1XX	X	High speed master code.
1111 1XX	X	Reserved
1111 0XX	X	10-bit slave addressing.

I²C does not restrict you from using these reserved addresses. However, if you use these reserved addresses, you may run into incompatibilities with other I²C components.

13.2.4.3 Transmitting and Receiving Protocol

The master can initiate data transmission and reception to/from the bus, acting as either a master-transmitter or master-receiver. A slave responds to requests from the master to either transmit data or receive data to/from the bus, acting as either a slave-transmitter or slave-receiver, respectively.

13.2.4.3.1 Master-Transmitter and Slave-Receiver

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

If the master-transmitter is transmitting data as shown in Fig 9-13, then the slave-receiver responds to the master-transmitter with an acknowledge pulse after every byte of data is received.

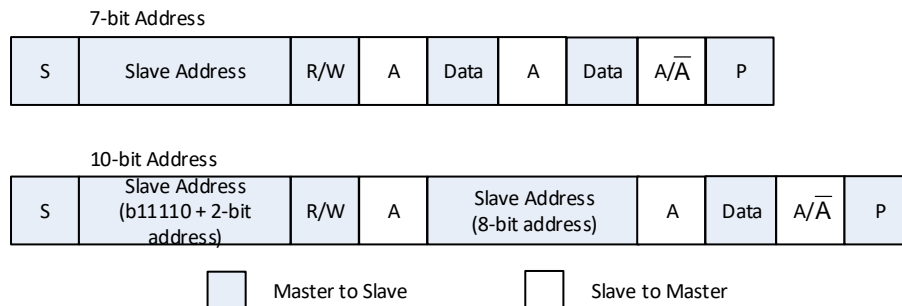


Fig 13-7 Master-Transmitter protocol

13.2.4.3.2 Master-Receiver and Slave-Transmitter

If the master is receiving data as shown in Fig 9-14, then the master responds to the slave-transmitter with an acknowledge pulse after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) so that the master can issue a STOP condition.

When a master does not want to relinquish the bus with a STOP condition, the master can issue a RESTART condition. This is identical to a START condition except it occurs after the ACK pulse. Operating in master mode, the I²C can then communicate with the same slave using a transfer of a different direction.

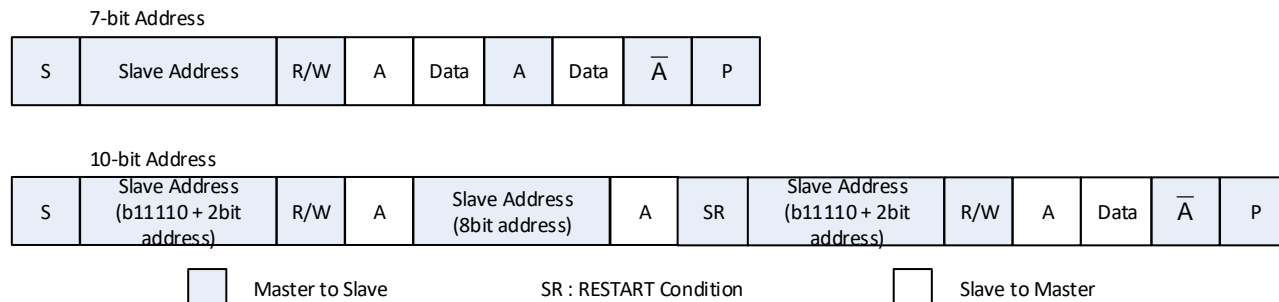


Fig 13-8 Master-Receiver protocol

13.2.4.4 START BYTE Transfer Protocol

The START BYTE transfer protocol is set up for systems that do not have an on-board dedicated I²C hardware module. When the I²C is addressed as a slave, it always samples the I²C bus at the highest speed supported so that it never requires a START BYTE transfer. However, when I²C is a master, it supports the generation of START BYTE transfers at the beginning of every transfer in case a slave device requires it. This protocol consists of seven zeros being transmitted followed by a 1, as illustrated in Fig 13-9. This allows the processor that is polling the bus to under-sample the address phase until 0 is detected. Once the microcontroller detects a 0, it switches from the under sampling rate to the correct rate of the master.

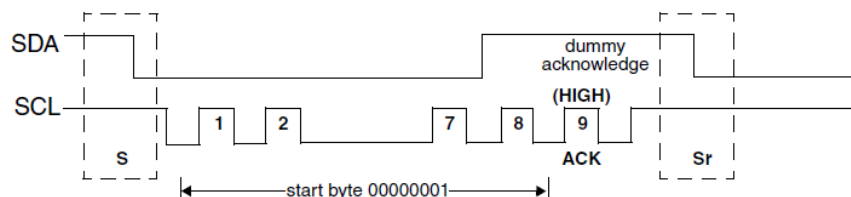


Fig 13-9 START BYTE transfer

The START BYTE procedure is as follows:

- (1) Master generates a START condition.
- (2) Master transmits the START byte (0000 0001).
- (3) Master transmits the ACK clock pulse. (Present only to conform with the byte handling format used on the bus)
- (4) No slave sets the ACK signal to 0.
- (5) Master generates a RESTART (R) condition.

A hardware receiver does not respond to the START BYTE because it is a reserved address and resets after the RESTART condition is generated.

13.2.4.5 General Call Transfer Protocol

The general call is for addressing every device connected to the I²C bus. However, if a device doesn't need any of the data supplied within the general call structure, it can ignore this address by not issuing an acknowledgement. If a device does require this address and behave as a slave-receiver. The second and following bytes will be acknowledged by every slave-receiver capable of handling this data. Fig 13-10 gives the General Call address format.

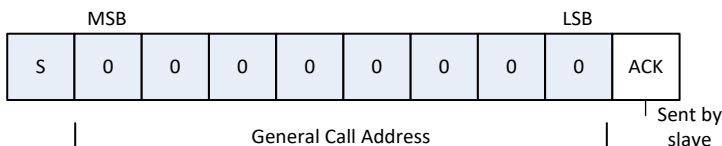


Fig 13-10 General call address format

When I²C acts as a slave, set IC_ACK_GENERAL_CALL (0x98) bit 0 to 1 to respond with a ACK when it receives a General Call, and a General Call interrupt will be issued. When set this bit to 0, the I²C does not generate General Call interrupts.

13.2.4.6 NULL DATA Transfer Protocol

NULL DATA transfer is used for some sensors. When IC_DATA_CMD[11] and IC_DATA_CMD[9] is set to 1, I²C would ignore REG_IC_TAR but take the TXFIFO data as slave address. It would only send TXFIFO data in address phase without any further transmission. Fig 13-11 gives the NULL DATA transfer format.

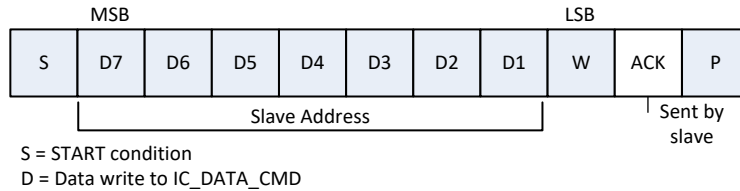


Fig 13-11 NULL DATA transfer format

13.2.5 Tx FIFO Management and START, STOP and RESTART Generation

Ameba-D I²C does not generate a STOP if the Tx FIFO becomes empty; in this situation the component holds the SCL line low, stalling the bus until a new entry is available in the Tx FIFO. A STOP condition is generated only when the user specifically requests it by setting bit 9 (Stop bit) of the command written to REG_IC_DATA_CMD register. Fig 13-12 shows the fields in IC_DATA_CMD. Please refer to the register description for more detail information.

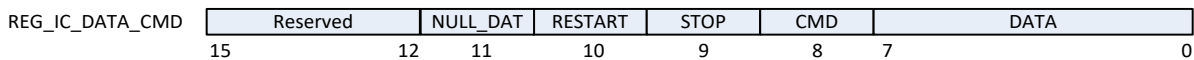


Fig 13-12 IC_DATA_CMD register content

Fig 13-13 illustrates the behavior of the I²C when the Tx FIFO becomes empty while operating as a master transmitter, as well as showing the generation of a STOP condition.

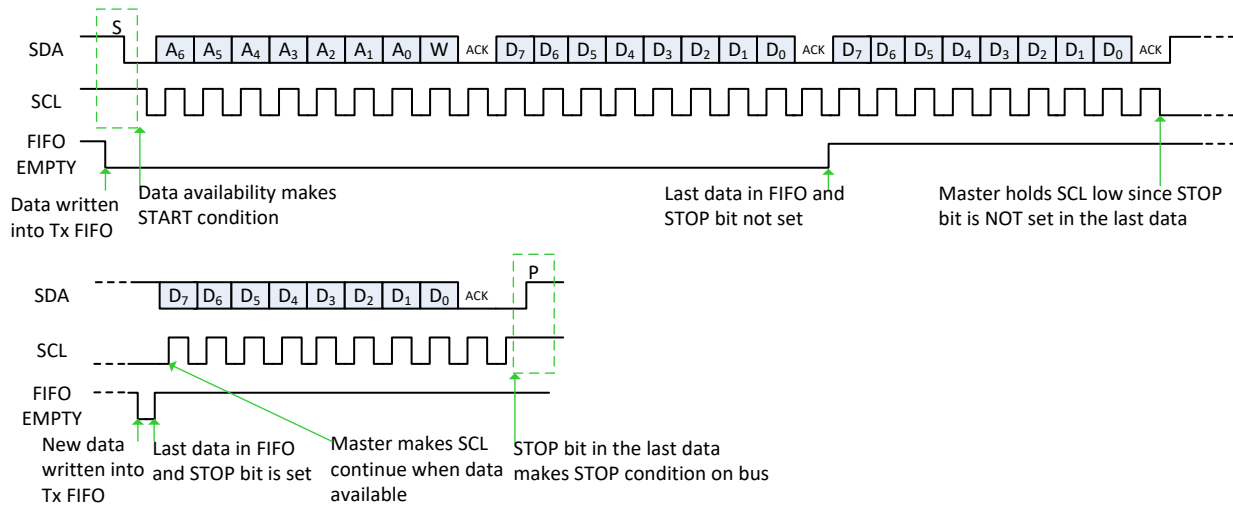


Fig 13-13 Master transmitter — Tx FIFO empties/STOP generation

Fig 13-14 illustrates the behavior of the I²C when the Tx FIFO becomes empty while operating as a master receiver, as well as showing the generation of a STOP condition.

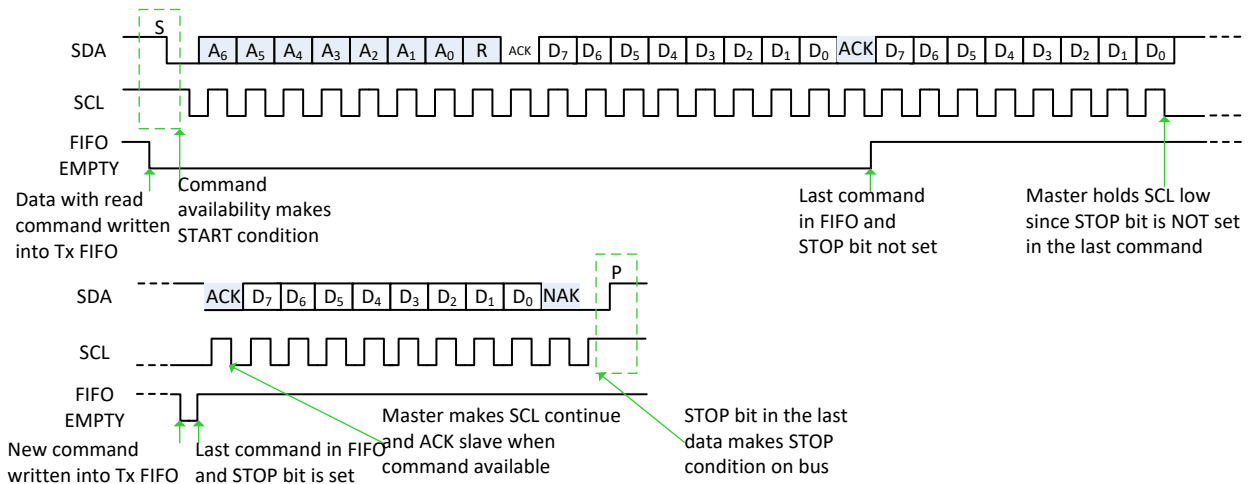


Fig 13-14 Master receiver — Tx FIFO empties/STOP generation

Fig 13-15 and Fig 13-16 illustrate configurations where the user can control the generation of RESTART conditions on the I²C bus. If bit 10 (Restart) of the IC_DATA_CMD register is set and the restart capability is enabled (IC_RESTART_EN=1), a RESTART is generated before the data byte is written to or read from the slave. If the restart capability is not enabled a STOP followed by a START is generated in place of the RESTART. Fig 13-15 illustrates this situation during operation as a master transmitter.

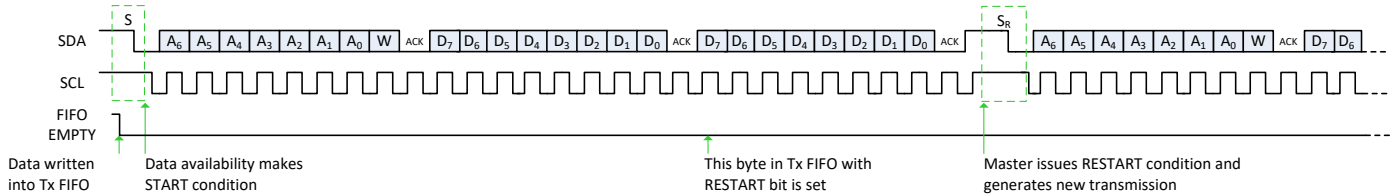


Fig 13-15 Master transmitter — Restart bit of IC_DATA_CMD is set

Fig 13-16 illustrates the same situation, but during operation as a master receiver.

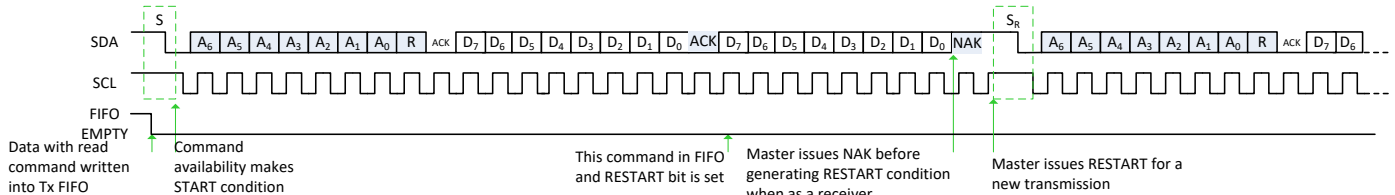


Fig 13-16 Master receiver — Restart bit of IC_DATA_CMD is set

Fig 13-17 illustrates operation as a master transmitter where the Stop bit of the IC_DATA_CMD register is set and the Tx FIFO is not empty.

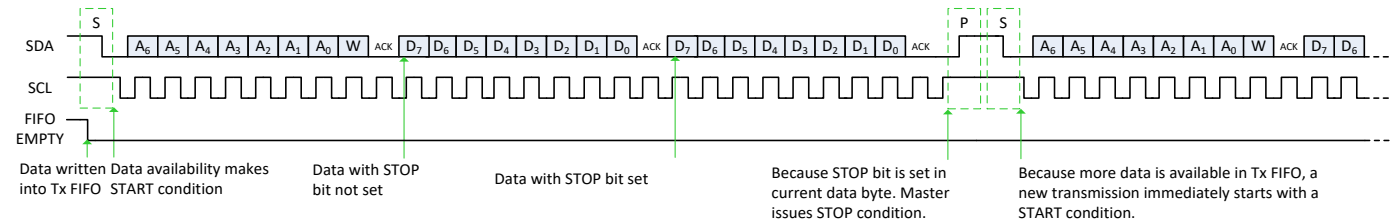


Fig 13-17 Master transmitter — Stop bit of IC_DATA_CMD set/Tx FIFO not empty

Fig 13-18 illustrates operation as a master receiver where the Stop bit of the IC_DATA_CMD register is set and the Tx FIFO is not empty.

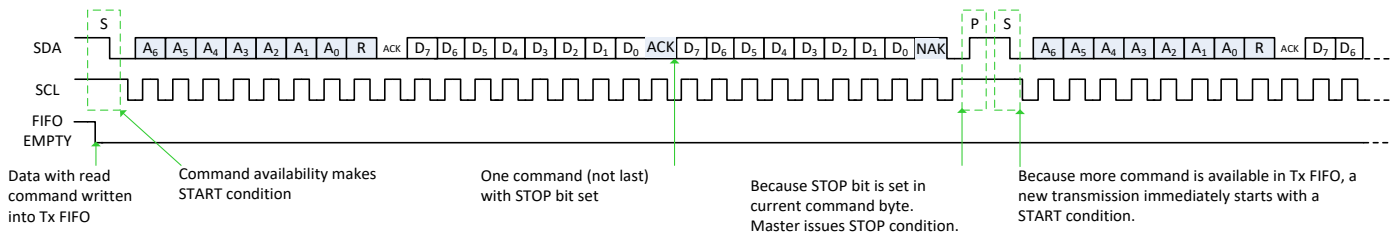


Fig 13-18 Master receiver — Stop bit of IC_DATA_CMD set/Tx FIFO not empty

13.2.6 Multiple Master Arbitration

The I²C bus protocol allows multiple masters to reside on the same bus. If there are two masters on the same I²C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating a START condition at the same time. Once a master (for example, a microcontroller) has control of the bus, no other master can take control until the first master sends a STOP condition and places the bus in an idle state.

Arbitration takes place on the SDA line, while the SCL line is 1. The master, which transmits a 1 while the other master transmits 0, loses arbitration and turns off its data output stage. The master that lost arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase.

Upon detecting that it has lost arbitration to another master, the I²C will stop generating SCL.

Fig 13-19 illustrates the timing of when two masters are arbitrating on the bus.

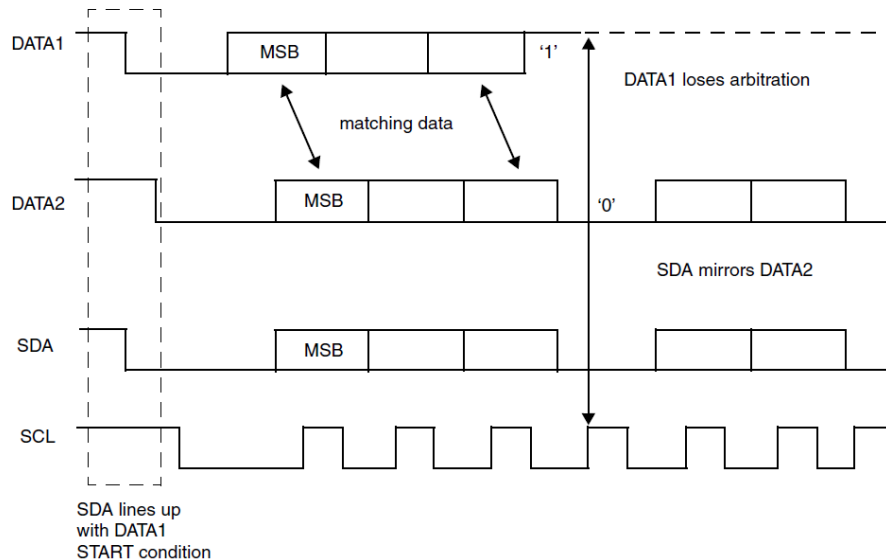


Fig 13-19 Multiple master arbitration

For high-speed mode, the arbitration cannot go into the data phase because each master is programmed with a unique high-speed master code. This 8-bitcode is defined by the system designer and is set by writing to the IC_HS_MADDR (High Speed Master Mode Code Address) register. Because the codes are unique, only one master can win arbitration, which occurs by the end of the transmission of the high-speed master code. Control of the bus is determined by address or master code and data sent by competing masters, so there is no central master nor any order of priority on the bus.

Slaves are not involved in the arbitration process.

13.2.7 Clock Synchronization

When two or more masters try to transfer information on the bus at the same time, they must arbitrate and synchronize the SCL clock. All masters generate their own clock to transfer messages. Data is valid only during the high period of SCL clock. Clock synchronization is performed using the wired-AND connection to the SCL signal. When the master transitions the SCL clock to 0, the master starts counting the low time of the SCL clock and transitions the SCL clock signal to 1 at the beginning of the next clock period. However, if another master is holding the SCL line to 0, then the master goes into a HIGH wait state until the SCL clock line transitions to 1.

All masters then count off their high time and the master with the shortest high time transitions the SCL line to 0. The masters then count out their low time and the one with the longest low time forces the other master into a HIGH wait state. Therefore, a synchronized SCL clock is generated, which is illustrated in Fig 13-20. Optionally, slaves may hold the SCL line low to slow down the timing on the I²C bus.

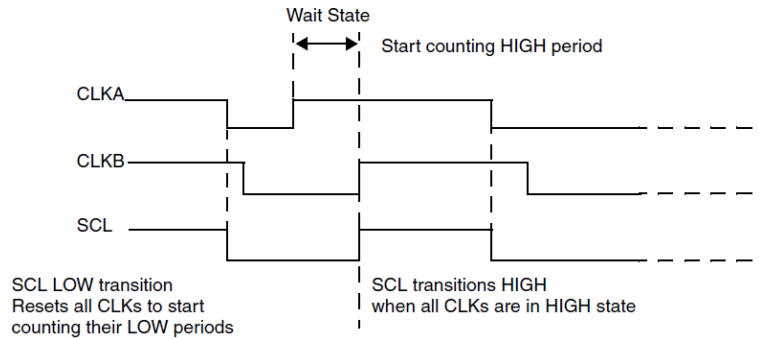


Fig 13-20 Multi-Master clock synchronization

13.2.8 Operation Modes

This section provides information on operation modes.

Note: The I²C should only be set to operate as an I²C Master, or I²C Slave, but not both simultaneously. This is achieved by ensuring that bit[6] (IC_SLAVE_DISABLE) and bit[0] (MASTER_MODE) of the IC_CON register are never set to 0 and 1 respectively.

13.2.8.1 Slave Mode Operation

This section discusses slave mode procedures.

13.2.8.1.1 Initial Configuration

To use the I²C as a slave, perform the following steps:

- (1) Disable the I²C by writing a '0' to bit 0 of the IC_ENABLE register.
- (2) Write to the IC_SAR register (bits 9:0) to set the slave address. This is the address to which the I²C responds.
- (3) Write to the IC_CON register to specify which type of addressing is supported (7- or 10-bit by setting bit 3). Enable the I²C in slave-only mode by writing a '0' into bit 6 (IC_SLAVE_DISABLE) and a '0' to bit 0 (MASTER_MODE).

Note: Slaves and masters do not have to be programmed with the same type of addressing 7- or 10-bit address. For instance, a slave can be programmed with 7-bit addressing and a master with 10-bit addressing, and vice versa.

- (4) Enable the I²C by writing a '1' in bit 0 of the IC_ENABLE register.

13.2.8.1.2 Slave-Transmitter Operation for A Single Byte

When another I²C master device on the bus addresses the I²C and requests data, the I²C acts as a slave-transmitter and the following steps occur:

- (1) The other I²C master device initiates an I²C transfer with an address that matches the slave address in the IC_SAR register of the I²C.
- (2) The I²C acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter.
- (3) The I²C asserts the RD_REQ interrupt (bit 5 of the IC_RAW_INTR_STAT register) and holds the SCL line low. It is in a wait state until software responds.

- a) Reads that indicate IC_RAW_INTR_STAT[5] (R_RD_REQ bit field) being set to 1 must be treated as the equivalent of the RD_REQ interrupt being asserted.
- b) Software must then act to satisfy the I²C transfer.
- (4) If there is any data remaining in the Tx FIFO before receiving the read request, then the I²C asserts a TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register) to flush the old data from the Tx FIFO.
Note: Because the I²C's Tx FIFO is forced into a flushed/reset state whenever a TX_ABRT Interrupt clear event occurs, it is necessary for software to release the I²C from this state by reading the IC_CLR_TX_ABRT register before attempting to write into the Tx FIFO. See register IC_RAW_INTR_STAT for more details.
 - a) Reads that indicate bit 6 (R_TX_ABRT) being set to 1 must be treated as the equivalent of the TX_ABRT interrupt being asserted.
 - b) There is no further action required from software.
- (5) Software writes to the IC_DATA_CMD register with the data to be written (by writing a '0' in bit 8).
- (6) Software must clear the RD_REQ and TX_ABRT interrupts (bits 5 and 6, respectively) of the IC_RAW_INTR_STAT register before proceeding.
- (7) The I²C releases the SCL and transmits the byte.
- (8) The master may hold the I²C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

13.2.8.1.3 Slave-Receiver Operation for A Single Byte

When another I²C master device on the bus addresses the I²C and is sending data, the I²C acts as a slave-receiver and the following steps occur:

- (1) The other I²C master device initiates an I²C transfer with an address that matches the I²C's slave address in the IC_SAR register.
- (2) The I²C acknowledges the sent address and recognizes the direction of the transfer to indicate that the I²C is acting as a slave-receiver.
- (3) I²C receives the transmitted byte and places it in the receive buffer.
Note: If the Rx FIFO is completely filled with data when a byte is pushed, then an overflow occurs and the I²C continues with subsequent I²C transfers. Because a NACK is not generated, software must recognize the overflow when indicated by the I²C (by the R_RX_OVER bit in the IC_INTR_STAT register) and take appropriate actions to recover from lost data. Hence, there is a real time constraint on software to service the Rx FIFO before the latter overflow as there is no way to reapply pressure to the remote transmitting master. You must select a deep enough Rx FIFO depth to satisfy the interrupt service interval of their system.
- (4) I²C asserts the RX_FULL interrupt (IC_RAW_INTR_STAT[2] register).
 If the RX_FULL interrupt has been masked, due to setting IC_INTR_MASK[2] register to 0 or setting IC_TX_TL to a value larger than 0, then it is recommended that a timing routine be implemented for periodic reads of the IC_STATUS register. Reads of the IC_STATUS register, with bit 3 (RFNE) set at 1, must then be treated by software as the equivalent of the RX_FULL interrupt being asserted.
- (5) Software may read the byte from the IC_DATA_CMD register (bits 7:0).
- (6) The other master device may hold the I²C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

13.2.8.1.4 Slave-Transfer Operation for Bulk Transfers

In the standard I²C protocol, all transactions are single byte transactions and the programmer responds to a remote master read request by writing one byte into the slave's Tx FIFO. When a slave (slave-transmitter) is issued with a read request (RD_REQ) from the remote master (master-receiver), at a minimum there should be at least one entry placed into the slave-transmitter's Tx FIFO. I²C is designed to handle more data in the Tx FIFO so that subsequent read requests can take that data without raising an interrupt to get more data. Ultimately, this eliminates the possibility of significant latencies being incurred between raising the interrupt for data each time had there been a restriction of having only one entry placed in the Tx FIFO.

This mode only occurs when I²C is acting as a slave-transmitter. If the remote master acknowledges the data sent by the slave-transmitter and there is no data in the slave's Tx FIFO, the I²C holds the I²C SCL line low while it raises the read request interrupt (RD_REQ) and waits for data to be written into the Tx FIFO before it can be sent to the remote master.

If the RD_REQ interrupt is masked, due to bit 5 (M_RD_REQ) of the IC_INTR_STAT register being set to 0, then it is recommended that a timing routine be used to activate periodic reads of the IC_RAW_INTR_STAT register. Reads of IC_RAW_INTR_STAT that return bit 5 (R_RD_REQ) set to 1 must be treated as the equivalent of the RD_REQ interrupt referred to in this section.

The RD_REQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt service handling routine (ISR). The ISR allows you to either write 1 byte or more than 1 byte into the Tx FIFO. During the transmission of these bytes to the master, if the master acknowledges the last byte, then the slave must raise the RD_REQ again because the master is requesting for more data.

If the programmer knows in advance that the remote master is requesting a packet of n bytes, then when another master addresses I²C and requests data, the Tx FIFO could be written with n number bytes and the remote master receives it as a continuous stream of data. For

example, the I²C slave continues to send data to the remote master as long as the remote master is acknowledging the data sent and there is data available in the Tx FIFO. There is no need to hold the SCL line low or to issue RD_REQ again.

If the remote master is to receive n bytes from the I²C but the programmer wrote a number of bytes larger than n to the Tx FIFO, then when the slave finishes sending the requested n bytes, it clears the Tx FIFO and ignores any excess bytes.

13.2.8.2 Master Mode Operation

This section discusses master mode procedures.

13.2.8.2.1 Initial Configuration

The target address and address format can be changed dynamically without having to disable I²C. This parameter only applies to when I²C is acting as a master because the slave requires the component to be disabled before any changes can be made to the address.

The procedures are very similar and are only different with regard to where the IC_10BITADDR_MASTER bit is set (bit 12 of IC_TAR register).

- (1) Disable the I²C by writing 0 to the IC_ENABLE register.
- (2) Write to the IC_CON register to set the maximum speed mode supported for slave operation (bits 2:1). Writing 1 to bit 0 and bit 6 to enable Master Module and disable Slave Module.
- (3) Write to the IC_TAR register the address of the I²C device to be addressed. It also indicates whether a General Call or a START BYTE command is going to be performed by I²C. The desired speed of the I²C master-initiated transfers, either 7-bit or 10-bit addressing, is controlled by the IC_10BITADDR_MASTER bit field (bit 12).
- (4) Enable the I²C by writing a '1' in the IC_ENABLE register.
- (5) Now write transfer direction and data to be sent to the IC_DATA_CMD register. If the IC_DATA_CMD register is written before the I²C is enabled, the data and commands are lost as the buffers are kept cleared when I²C is disabled.

13.2.8.2.2 Dynamic IC_TAR or IC_10BITADDR_MASTER Update

The I²C supports dynamic updating of the IC_TAR (bits 9:0) and IC_10BITADDR_MASTER (bit 12) bit fields of the IC_TAR register. You can dynamically write to the IC_TAR register provided the following conditions are met:

- (1) I²C is not enabled (IC_ENABLE=0); OR
- (2) I²C is enabled (IC_ENABLE=1); AND
I²C is NOT engaged in any Master (tx, rx) operation (IC_STATUS[5]=0); AND
I²C is enabled to operate in Master mode (IC_CON[0]=1); AND
there are NO entries in the Tx FIFO (IC_STATUS[2]=1)

13.2.8.2.3 Master Transmit and Master Receive

The I²C supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I²C Rx/Tx Data Buffer and Command Register (IC_DATA_CMD). The bit 8 (CMD) should be written to 0 for I²C write operations. Subsequently, a read command may be issued by writing "don't cares" to the lower byte of the IC_DATA_CMD register, and a 1 should be written to the CMD bit. The I²C master continues to initiate transfers as long as there are commands present in the transmit FIFO. If the STOP bit set 1, the I²C inserts a STOP condition after completing the current transfer.

13.2.9 IC_CLK Frequency Configuration

When the I²C is configured as a master, the *CNT registers must be set before any I²C bus transaction can take place in order to ensure proper I/O timing. The *CNT registers are:

- IC_SS_SCL_HCNT
- IC_SS_SCL_LCNT
- IC_FS_SCL_HCNT
- IC_FS_SCL_LCNT
- IC_HS_SCL_HCNT
- IC_HS_SCL_LCNT

Note: It is not necessary to program any of the *CNT registers if the I²C is enabled to operate only as an I²C slave, since these registers are used only to determine the SCL timing requirements for operation as an I²C master.

13.2.9.1 Minimum High and Low Counts

When the I²C operates as an I²C master, in both transmit and receive transfers:

- Minimum value that can be programmed in the *_LCNT registers is 8
- Minimum value allowed for the *_HCNT registers is 6

The minimum value of 8 for the *_LCNT registers is due to the time required for the I²C to drive SDA after a negative edge of SCL.

The minimum value of 6 for the *_HCNT register is due to the time required for the I²C to sample SDA during the high period of SCL.

13.2.9.2 Programmable Duty Cycle

This section describes the minimum ic_clk frequencies that the I²C supports for each speed mode, and the associated high and low count values. It should be noted that these limits apply to the I²C in both master and slave modes. The limits for slave mode are required so that the I²C does not break the Thd; dat maximum I²C protocol timing requirement.

13.2.9.2.1 Calculating High and Low Counts

The calculations below show how to calculate SCL high and low counts for each speed mode in the I²C.

Following conditions should be given:

- Ic_clk frequency: fic_clk → IC_CLK_PERIOD=1/ fic_clk
- Desired duty cycle: HIGH_TIME : LOW_TIME
- SCL frequency: fSCL → SCL_PERIOD=1/ fSCL

While we want to get:

- HIGH_COUNT & LOW_COUNT

We can derive the equations:

$$\begin{cases} \frac{SCL_PERIOD}{HIGH_COUNT + LOW_COUNT} = IC_CLK_PERIOD & (1) \\ \frac{HIGH_COUNT}{HIGH_TIME} = \frac{LOW_COUNT}{LOW_TIME} & (2) \end{cases}$$

Then we can get:

$$\begin{cases} HIGH_COUNT = \frac{HIGH_TIME * SCL_PERIOD}{(HIGH_TIME + LOW_TIME) * IC_CLK_PERIOD} \\ LOW_COUNT = \frac{LOW_TIME * SCL_PERIOD}{(HIGH_TIME + LOW_TIME) * IC_CLK_PERIOD} \end{cases}$$

Take Ameba-D I²C as an example:

- Ic_clk frequency: 10M → IC_CLK_PERIOD=100ns
- Desired duty cycle:
 - 100K: 40:47
 - 400K: 6:13
- SCL frequency:
 - 100K → SCL_PERIOD=10000ns
 - 400K → SCL_PERIOD=2500ns

Standard Speed Mode (100K):

- IC_SS_SCL_HCNT = $\frac{40 * 10000}{(40 + 47) * 100} = 45$
- IC_SS_SCL_LCNT = $\frac{47 * 10000}{(40 + 47) * 100} = 54$

Fast Speed Mode (400K):

- $IC_FS_SCL_HCNT = \frac{6*2500}{(6+13)*100} = 7$
- $IC_FS_SCL_LCNT = \frac{13*2500}{(6+13)*100} = 17$

Note: Since slave may stretch SCL line, the actual speed is a little bit smaller than 100K/400K if we set the above calculated values. In order to reach the accurate speed, calculated values should be reduced a little according to test results.

13.2.10 Programmable SDA Hold Time

As each application will encounter differing board delays, I²C contains a software programmable register IC_SDA_HOLD (0x7C) to enable dynamic adjustment of the SDA hold time.

When I²C acts as a master transmitter, the hold time is 6 ic_clk periods if IC_SDA_HOLD is smaller than or equal to 6; the hold time is equal to value of IC_SDA_HOLD ic_clk periods if IC_SDA_HOLD is larger than 6.

When I²C acts as a slave transmitter, the hold time is IC_SDA_HOLD + 5 to IC_SDA_HOLD + 6 ic_clk periods.

Note: The 8th bit of data is not under control of IC_SDA_HOLD, since after 8th SCL clock negative edge, receiver controls the SDA line to respond ACK or NACK.

The IC_SDA_HOLD register can be programmed only when I²C is disabled (IC_ENABLE=0).

13.2.11 DMA Controller Interface

There are two DMA modes of I²C discussed in the following sections:

- 16-bit FIFO (DMA legacy)
- 8-bit FIFO with transfer control register.

13.2.11.1 DMA Legacy Control Mode

The Ameba-D I²C has an optional built-in DMA capability; it has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA.

13.2.11.1.1 Enabling the DMA Controller Interface

To enable the DMA Controller interface on the Ameba-D I²C, you must write the DMA Control Register (IC_DMA_CR). Writing a 1 into the TDMAE bit field of IC_DMA_CR register enables the I²C transmit handshaking interface. Writing a 1 into the RDMAE bit field of the IC_DMA_CR register enables the I²C receive handshaking interface.

13.2.11.1.2 Transmit Watermark Level Select

During I²C serial transfers, transmit FIFO requests are made to the DMA Controller whenever the number of entries in the transmit FIFO is less than or equal to the DMA Transmit Data Level Register (IC_DMA_TDLR) value; this is known as the watermark level. The DMA Controller responds by writing a burst of data to the transmit FIFO buffer.

Data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously; that is, when the FIFO begins to empty another DMA request should be triggered. Otherwise, the FIFO will run out of data causing a STOP to be inserted on the I²C bus. To prevent this condition, the user must set the watermark level correctly.

The goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. For optimal operation, DMA.CTLx.DEST_MSIZEx should be set at the FIFO level that triggers a transmit DMA request; that is:

$$DMA.CTLx.DEST_MSIZEx = I2C.FIFO_DEPTH - I2C.IC_DMA_TDLR$$

13.2.11.1.3 Receive Watermark Level Select

During I²C serial transfers, receive FIFO requests are made to the DMA Controller whenever the number of entries in the receive FIFO is at or above the DMA Receive Data Level Register(IC_DMA_RDLR); that is, IC_DMA_RDLR+1. This is known as the watermark level. The DMA Controller responds by fetching a burst of data from the receive FIFO buffer of length CTLx.SRC_MSIZEx.

Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously; that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise, the FIFO will fill with data (overflow). To prevent this condition, the user must correctly set the watermark level.

Similar to choosing the transmit watermark level described earlier, the receive watermark level, IC_DMA_RDLR+1, should be set to minimize the probability of overflow. For optimal operation, DMA.CTLx.SRC_MSIZEx should be set at the watermark level; that is:

$$\text{DMA.CTLx.SRC_MSIZE} = \text{I2C.IC_DMA_RDLR} + 1$$

13.2.11.2 8-Bit FIFO with DMA Transfer Control Register

For 8-bit FIFO mode, two methods are provided for different use scenario: 8-bit FIFO with DMA transfer control register and 8-bit FIFO with DMA transfer descriptor. In these methods, I²C must co-operate with GDMA channels. To change DMA mode, a DMA mode register is added. Software should set a correct value to DMA mode field before any further setup. Both Master and Slave mode can use this DMA mode.

In 8-bit FIFO with DMA transfer control register mode, transfer process is controlled by two additional register: IC_DMA_CMD and IC_DMA_DATA_LEN register. Besides these registers, a significant difference from normal I²C DMA operation is that software doesn't need to fill dummy read command into TXFIFO of I²C. Since transfer data length could be given in DMA transfer data length register.

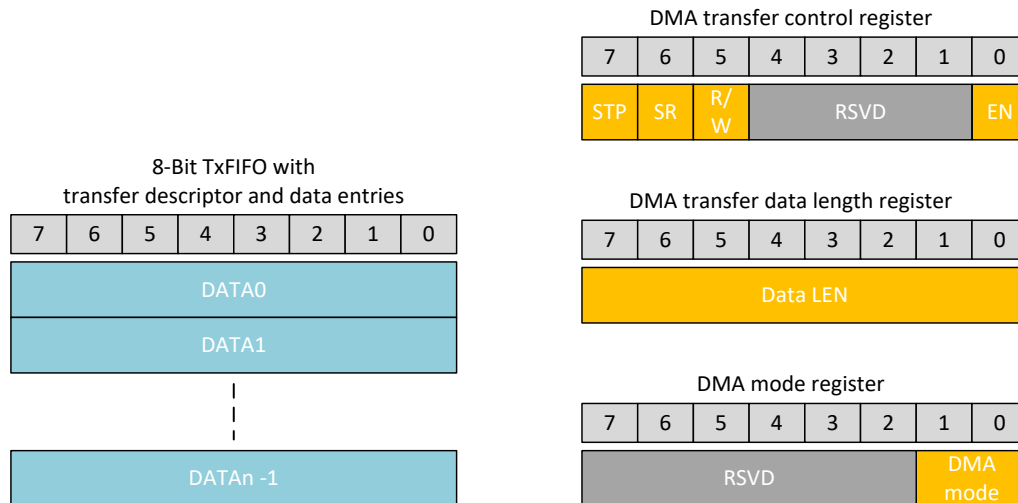


Fig 13-21 I²C 8-bit FIFO content with transfer control register

IC_DMA_CMD contains several fields: R/W is for transfer direction. SR is to drive a RESTART signal or not after the last data transferred. STP is to drive a STOP signal or not after the last data transferred.

13.2.12 Low Power Mode

The Ameba-D provides two sleep modes to reduce power consumption:

- sleep mode stops the processor clock
- deep sleep mode stops the system clock and switches off the PLL and flash memory

To sleep mode, Ameba-D provides many wakeup source, such as Timer, GPIO, RTC, I²C. For I²C transmission, master device need to dominate the transfer, so it can't sleep. Slave device can sleep and be wakeup when the address sent by master matching its address. Ameba-D I²C also supports low power mode to reduce power consumption.

13.3 Registers

This section describes the programmable registers of the I²C.

13.3.1 Register Memory Map

Table 13-2 lists the details of I²C register memory map. The base address of I²C0 is 0x4800_C000, and the size is 1K.

Note: A read operation to an address location that contains unused bits results in a 0 value being returned on each of the unused bits.

Table 13-2 I²C Memory map

Name	Address Offset	Access	Description
IC_CON	0x00	R/W	I ² C Control Register
IC_TAR	0x04	R/W	I ² C Target Address Register
IC_SAR	0x08	R/W	I ² C Slave Address Register
IC_HS_MADDR	0x0C	R/W	I ² C High Speed Master Mode Code Address Register
IC_DATA_CMD	0x10	R/W	I ² C Rx/Tx Data Buffer and Command Register
IC_SS_SCL_HCNT	0x14	R/W	Standard Speed I ² C Clock SCL High Count Register
IC_SS_SCL_LCNT	0x18	R/W	Standard Speed I ² C Clock SCL Low Count Register
IC_FS_SCL_HCNT	0x1C	R/W	Fast Speed I ² C Clock SCL High Count Register
IC_FS_SCL_LCNT	0x20	R/W	Fast Speed I ² C Clock SCL Low Count Register
IC_HS_SCL_HCNT	0x24	R/W	High Speed I ² C Clock SCL High Count Register
IC_HS_SCL_LCNT	0x28	R/W	High Speed I ² C Clock SCL Low Count Register
IC_INTR_STAT	0x2C	R	I ² C Interrupt Status Register
IC_INTR_MASK	0x30	R/W	I ² C Interrupt Mask Register
IC_RAW_INTR_STAT	0x34	R	I ² C Raw Interrupt Status Register
IC_RX_TL	0x38	R/W	I ² C Receive FIFO Threshold Register
IC_TX_TL	0x3C	R/W	I ² C Transmit FIFO Threshold Register
IC_CLR_INTR	0x40	R	Clear Combined and Individual Interrupt Register
IC_CLR_RX_UNDER	0x44	R	Clear RX_UNDER Interrupt Register
IC_CLR_RX_OVER	0x48	R	Clear RX_OVER Interrupt Register
IC_CLR_TX_OVER	0x4C	R	Clear TX_OVER Interrupt Register
IC_CLR_RD_REQ	0x50	R	Clear RD_REQ Interrupt Register
IC_CLR_TX_ABRT	0x54	R	Clear TX_ABRT Interrupt Register
IC_CLR_RX_DONE	0x58	R	Clear RX_DONE Interrupt Register
IC_CLR_ACTIVITY	0x5C	R	Clear ACTIVITY Interrupt Register
IC_CLR_STOP_DET	0x60	R	Clear STOP_DET Interrupt Register
IC_CLR_START_DET	0x64	R	Clear START_DET Interrupt Register
IC_CLR_GEN_CALL	0x68	R	Clear GEN_CALL Interrupt Register
IC_ENABLE	0x6C	R/W	I ² C Enable Register
IC_STATUS	0x70	R	I ² C Status Register
IC_TXFLR	0x74	R	I ² C Transmit FIFO Level Register
IC_RXFLR	0x78	R	I ² C Receive FIFO Level Register
IC_SDA_HOLD	0x7C	R/W	SDA Hold Time Length Register
IC_TX_ABRT_SOURCE	0x80	R	I ² C Transmit Abort Status Register
IC_SLV_DATA_NACK_ONLY	0x84	R/W	Generate Slave Data NACK Register
IC_DMA_CR	0x88	R/W	DMA Control Register
IC_DMA_TDLR	0x8C	R/W	DMA Transmit Data Level Register
IC_DMA_RDLR	0x90	R/W	DMA Receive Data Level Register
IC_SDA_SETUP	0x94	R/W	I ² C SDA Setup Register
IC_ACK_GENERAL_CALL	0x98	R/W	I ² C ACK General Call Register
IC_ENABLE_STATUS	0x9C	R	I ² C Enable Status Register
IC_DMA_CMD	0xA0	R/W	I ² C DMA Command Register

IC_DMA_DATA_LEN	0xA4	R/W	I ² C DMA Mode Transfer Data Length Register
IC_DMA_MODE	0xA8	R/W	I ² C DMA Mode Register
IC_SLEEP	0xAC	R/W	I ² C Sleep Mode Register
IC_CLR_ADDR_MATCH	0xE4	R	Clear Slave Mode Address Match Interrupt Register
IC_CLR_DMA_DONE	0xE8	R	Clear DMA DONE Interrupt Register
IC_FILTER	0xEC	R/W	I ² C Filter Register
IC_COMP_VER	0xFC	R	I ² C Component Version Register

13.3.2 Registers and Field Descriptions

The following sections describe the registers listed in Table 13-2.

13.3.2.1 IC_CON

- **Name:** I²C Control Register
- **Size:** 32 bits
- **Address offset:** 0x00
- **Read/write access:** read/write

This register can be written only when the I²C is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.

31	30	...	8	7	6	5	4	3	2	1	0
RSVD					IC_SLAVE_DISABLE	IC_RESTART_EN	RSVD	IC_10BITADDR_SLAVE	SPEED	MASTER_MODE	
					R/W	R/W		R/W	R/W	R/W	

Bit	Name	Access	Reset	Description
31:7	RSVD	N/A	-	Reserved
6	IC_SLAVE_DISABLE	R/W	0x0	<p>This bit controls whether I²C has its slave disabled, which means once the present signal is applied, then this bit takes on the value of the configuration parameter IC_SLAVE_DISABLE. You have the choice of having the slave enabled or disabled after reset is applied, which means software does not have to configure the slave. By default, the slave is always enabled (in reset state as well).</p> <p>If you need to disable it after reset, set this bit to 1. If this bit is set to 1 (slave is disabled), I²C functions only as a master and does not perform any action that requires a slave.</p> <ul style="list-style-type: none"> ● 0: Slave is enabled ● 1: Slave is disabled <p>Note: Software should ensure that if this bit is written with '0,' then bit[0] should also be written with a '0'.</p>
5	IC_RESTART_EN	R/W	0x0	<p>Determines whether RESTART conditions may be sent when acting as a master. Some older slaves do not support handling RESTART conditions; however, RESTART conditions are used in several I²C operations.</p> <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable <p>When the RESTART is disabled, the I²C master is incapable of performing the following functions:</p> <ul style="list-style-type: none"> ● Sending a START BYTE ● Performing any high-speed mode operation ● Performing direction changes in combined format mode ● Performing a read operation with a 10-bit address <p>By replacing RESTART condition followed by a STOP and a subsequent START condition, split operations are broken down into multiple I²C transfers. If the above operations are performed, it will result in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register.</p>
4	RSVD	N/A	-	Reserved

3	IC_10BITADDR_SLAVE	R/W	0x0	When acting as a slave, this bit controls whether the I ² C responds to 7- or 10-bit addresses. <ul style="list-style-type: none"> 0: 7-bit addressing. The I²C ignores transactions that involve 10-bit addressing; for 7-bit addressing, only the lower 7 bits of the IC_SAR register are compared. 1: 10-bit addressing. The I²C responds to only 10-bit addressing transfers that match the full 10 bits of the IC_SAR register.
2:1	SPEED	R/W	0x0	These bits control at which speed the I ² C operates; its setting is relevant only if one is operating the I ² C in master mode. Hardware protects against illegal values being programmed by software. <ul style="list-style-type: none"> 1: Standard mode (0 to 100Kbit/s) 2: Fast mode ($\leq 400\text{Kbit/s}$) 3: High speed mode ($\leq 3.4\text{Mbit/s}$)
0	MASTER_MODE	R/W	0x0	This bit controls whether the I ² C master is enabled. <ul style="list-style-type: none"> 0: Master is disabled 1: Master is enabled Note: Software should ensure that if this bit is written with '1,' then bit[6] should also be written with a '1'.

13.3.2.2 IC_TAR

- **Name:** I²C Target Address Register
- **Size:** 32 bits
- **Address offset:** 0x04
- **Read/write access:** read/write

Writes to IC_TAR succeed when one of the following conditions are true:

- I²C is NOT enabled (IC_ENABLE is set to 0); or I²C is enabled (IC_ENABLE is set to 1); AND
- I²C is NOT engaged in any master (tx, rx) operation (IC_STATUS[5]=0); AND I²C is enabled to operate in master mode (IC_CON[0]=1); AND there are NO entries in the Tx FIFO (IC_STATUS[2]=1)

31	30	...	14	13	12	11	10	9	8	...	1	0
RSVD					IC_10BITADDR_MASTER	SPECIAL	GC_OR_START	IC_TAR				
					R/W	R/W	R/W	R/W				

Bit	Name	Access	Reset	Description
31:13	RSVD	N/A	-	Reserved
12	IC_10BITADDR_MASTER	R/W	0x0	This bit controls whether the I ² C starts its transfers in 7-or 10-bit addressing mode when acting as a master. <ul style="list-style-type: none"> 0: 7-bit addressing 1: 10-bit addressing
11	SPECIAL	R/W	0x0	This bit indicates whether software performs a General Call or START BYTE command. <ul style="list-style-type: none"> 0: Ignore bit 10 (GC_OR_START) and use IC_TAR normally 1: Perform special I²C command as specified in bit 10 (GC_OR_START)
10	GC_OR_START	R/W	0x0	If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the I ² C. <ul style="list-style-type: none"> 0: General Call Address – after issuing a General Call, only writes may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register. The I²C remains in General Call mode until the SPECIAL bit (bit 11) value is cleared. 1: START BYTE
9:0	IC_TAR	R/W	0x10	This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits.

Note: It is not necessary to perform any write to this register if I²C is enabled as an I²C slave only.

13.3.2.3 IC_SAR

- **Name:** I²C Slave Address Register
- **Size:** 32 bits
- **Address offset:** 0x08
- **Read/write access:** read/write

31	30	29	...	12	11	10	9	8	7	...	2	1	0
RSVD										IC_SAR			
										R/W			

Bit	Name	Access	Reset	Description
31:10	RSVD	N/A	-	Reserved
9:0	IC_SAR	R/W	0x11	<p>The IC_SAR holds the slave address when the I²C is operating as a slave. For 7-bit addressing, only IC_SAR[6:0] is used.</p> <p>This register can be written only when the I²C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.</p> <p>Note: The default values cannot be any of the reserved address locations: that is, 0x00 to 0x07, or 0x78 to 0x7f. The correct operation of the device is not guaranteed if you program the IC_SAR or IC_TAR to a reserved value.</p>

Note: It is not necessary to perform any write to this register if I²C is enabled as an I²C master only.

13.3.2.4 IC_HS_MADDR

- **Name:** I²C High Speed Master Mode Code Address Register
- **Size:** 32 bits
- **Address offset:** 0x0C
- **Read/write access:** read/write

31	30	29	...	5	4	3	2	1	0
RSVD								IC_HS_MAR	
								R/W	

Bit	Name	Access	Reset	Description
31:3	RSVD	N/A	-	Reserved
2:0	IC_HS_MAR	R/W	0x0	<p>This bit field holds the value of the I²C HS mode master code. HS-mode master codes are reserved 8-bit codes (00001xxx) that are not used for slave addressing or other purposes. Each master has its unique master code; up to eight high speed mode masters can be present on the same I²C bus system. Valid values are from 0 to 7.</p> <p>This register can be written only when the I²C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.</p>

Note: It is not necessary to perform any write to this register if I²C is enabled as an I²C slave only.

13.3.2.5 IC_DATA_CMD

- **Name:** I²C Rx/Tx Data Buffer and Command Register
- **Size:** 32 bits
 - Bit[11:8] exists if DMA_MODE = 0
- **Address offset:** 0x10
- **Read/write access:** read/write

This is the register the CPU writes to when filling the Tx FIFO and the CPU reads from when retrieving bytes from Rx FIFO.

Note: In order for the I²C to continue acknowledging reads, a read command should be written for every byte that is to be received; otherwise the I²C will stop acknowledging.

31	30	...	13	12	11	10	9	8	7	6	...	1	0
RSVD					NULL_DATA	RESTART	STOP	CMD	DAT				
					R/W	R/W	R/W	R/W	R/W				

Bit	Name	Access	Reset	Description
31:12	RSVD	N/A	-	Reserved
11	NULL_DATA	R/W	0x0	This bit controls whether to transfer slave address only. When this bit is set to 1, I ² C would ignore REG_IC_TAR but take the TXFIFO data as slave address. It would only send TXFIFO data in address phase without any further transmission. <ul style="list-style-type: none"> 1: Send TXFIFO data as slave address. 0: Normal operation. This bit will NOT influence any transfer sequence.
10	RESTART	R/W	0x0	This bit controls whether a RESTART is issued before the byte is sent or received. <ul style="list-style-type: none"> 1: If IC_RESTART_EN is 1, a RESTART is issued before the data is sent/received (according to the value of CMD), regardless of whether or not the transfer direction is changing from the previous command. 0: If IC_RESTART_EN is 0, a STOP followed by a START is issued instead.
9	STOP	R/W	0x0	This bit controls whether a STOP is issued after the byte is sent or received. <ul style="list-style-type: none"> 1: STOP is issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus. 0: STOP is not issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master continues the current transfer by sending/receiving data bytes according to the value of the CMD bit. If the Tx FIFO is empty, the master holds the SCL line low and stalls the bus until a new command is available in the Tx FIFO.
8	CMD	R/W	0x0	This bit controls whether a read or a write is performed. This bit does not control the direction when the I ² C acts as a slave. It controls only the direction when it acts as a master. <ul style="list-style-type: none"> 1: Read 0: Write When a command is entered in the Tx FIFO, this bit distinguishes the write and read commands. In slave-receiver mode, this bit is a “don’t care” because writes to this register are not required. In slave-transmitter mode, a “0” indicates that the data in IC_DATA_CMD is to be transmitted.
7:0	DAT	R/W	0x0	This register contains the data to be transmitted or received on the I ² C bus. If you are writing to this register and want to perform a read, bits 7:0 (DAT) are ignored by the I ² C. However, when you read this register, these bits return the value of data received on the I ² C interface.

13.3.2.6 IC_SS_SCL_HCNT

- **Name:** Standard Speed I²C Clock SCL High Count Register
- **Size:** 32 bits
- **Address offset:** 0x14
- **Read/write access:** read/write

31	30	29	...	18	17	16	15	14	13	...	2	1	0
RSVD							IC_SS_SCL_HCNT						
							R/W						

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	IC_SS_SCL_HCNT	R/W	0x190	This register must be set before any I ² C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed. This register can be written only when the I ² C interface is disabled which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.

13.3.2.7 IC_SS_SCL_LCNT

- **Name:** Standard Speed I²C Clock SCL Low Count Register
- **Size:** 32 bits
- **Address offset:** 0x18
- **Read/write access:** read/write

31	30	29	...	18	17	16	15	14	13	...	2	1	0
RSVD							IC_SS_SCL_LCNT						
							R/W						

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	IC_SS_SCL_LCNT	R/W	0x1D6	This register must be set before any I ² C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low-period count for standard speed. This register can be written only when the I ² C interface is disabled which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.

13.3.2.8 IC_FS_SCL_HCNT

- **Name:** Fast Speed I²C Clock SCL High Count Register
- **Size:** 32 bits
- **Address offset:** 0x1C
- **Read/write access:** read/write

31	30	29	...	18	17	16	15	14	13	...	2	1	0
RSVD							IC_FS_SCL_HCNT						
							R/W						

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	IC_FS_SCL_HCNT	R/W	0x3C	This register must be set before any I ² C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast speed. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. This register can be written only when the I ² C interface is disabled which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.

13.3.2.9 IC_FS_SCL_LCNT

- **Name:** Fast Speed I²C Clock SCL Low Count Register
- **Size:** 32 bits
- **Address offset:** 0x20
- **Read/write access:** read/write

31	30	29	...	18	17	16	15	14	13	...	2	1	0
RSVD							IC_FS_SCL_LCNT						
							R/W						

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	IC_FS_SCL_LCNT	R/W	0x82	This register must be set before any I ² C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low-period count for fast speed. It is used in high speed mode to send the Master Code and START BYTE or General CALL. This register can be written only when the I ² C interface is disabled which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.

13.3.2.10 IC_HS_SCL_HCNT

- **Name:** High Speed I²C Clock SCL High Count Register
- **Size:** 32 bits
- **Address offset:** 0x24
- **Read/write access:** read/write

31	30	29	...	18	17	16	15	14	13	...	2	1	0
RSVD							IC_HS_SCL_HCNT						
							R/W						

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	IC_HS_SCL_HCNT	R/W	0x6	This register must be set before any I ² C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for high speed. The SCL High time depends on the loading of the bus. For 100pF loading, the SCL High time is 60ns; for 400pF loading, the SCL High time is 120ns. This register can be written only when the I ² C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.

13.3.2.11 IC_HS_SCL_LCNT

- **Name:** High Speed I²C Clock SCL Low Count Register
- **Size:** 32 bits
- **Address offset:** 0x28
- **Read/write access:** read/write

31	30	29	...	18	17	16	15	14	13	...	2	1	0
RSVD							IC_HS_SCL_LCNT						
							R/W						

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	IC_HS_SCL_LCNT	R/W	0x10	This register must be set before any I ² C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low-period count for high speed. The SCL low time depends on the loading of the bus. For 100pF loading, the SCL low time is 160ns; for 400pF loading, the SCL low time is 320ns. This register can be written only when the I ² C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.

13.3.2.12 IC_INTR_STAT

- **Name:** I²C Interrupt Status Register
- **Size:** 32 bits
- **Address offset:** 0x2C
- **Read/write access:** read-only

Each bit in this register has a corresponding mask bit in the IC_INTR_MASK register. These bits are cleared by reading the matching interrupt clear register. The unmasked raw versions of these bits are available in the IC_RAW_INTR_STAT register.

31	30	29	28	27	26	25	24
RSVD							
23	22	21	20	19	18	17	16
RSVD							
15	14	13	12	11	10	9	8
R_DMA_I2C_DONE	R_MS_CODE_DET	RSVD	R_ADDR_MATCH	R_GEN_CALL	R_START_DET	R_STOP_DET	R_ACTIVITY

R	R		R	R	R	R	R
7	6	5	4	3	2	1	0
R_RX_DONE	R_TX_ABRT	R_RD_REQ	R_TX_EMPTY	R_TX_OVER	R_RX_FULL	R_RX_OVER	R_RX_UNDER
R	R	R	R	R	R	R	R

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15	R_DMA_I2C_DONE	R	0x0	See IC_RAW_INTR_STAT for a detailed description of these bits.
14	R_MS_CODE_DET	R	0x0	
13	RSVD	N/A	-	
12	R_ADDR_MATCH	R	0x0	
11	R_GEN_CALL	R	0x0	
10	R_START_DET	R	0x0	
9	R_STOP_DET	R	0x0	
8	R_ACTIVITY	R	0x0	
7	R_RX_DONE	R	0x0	
6	R_TX_ABRT	R	0x0	
5	R_RD_REQ	R	0x0	
4	R_TX_EMPTY	R	0x0	
3	R_TX_OVER	R	0x0	
2	R_RX_FULL	R	0x0	
1	R_RX_OVER	R	0x0	
0	R_RX_UNDER	R	0x0	

13.3.2.13 IC_INTR_MASK

- **Name:** I²C Interrupt Mask Register
- **Size:** 32 bits
- **Address offset:** 0x30
- **Read/write access:** read/write

These bits mask their corresponding interrupt status bits. This register is active low; a value of 0 masks the interrupt, whereas a value of 1 unmask the interrupt.

31	30	29	28	27	26	25	24
RSVD							
23	22	21	20	19	18	17	16
RSVD							
15	14	13	12	11	10	9	8
M_DMA_I2C_DONE	M_MS_CODE_DET	RSVD	M_ADDR_MATCH	M_GEN_CALL	M_START_DET	M_STOP_DET	M_ACTIVITY
R/W	R/W		R/W	R/W	R/W	R/W	R/W
7	6	5	4	3	2	1	0
M_RX_DONE	M_TX_ABRT	M_RD_REQ	M_TX_EMPTY	M_TX_OVER	M_RX_FULL	M_RX_OVER	M_RX_UNDER
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15	M_DMA_I2C_DONE	R/W	0x0	These bits mask their corresponding interrupt status bits in the IC_INTR_STAT register.
14	M_MS_CODE_DET	R/W	0x0	
13	RSVD	N/A	-	
12	M_ADDR_MATCH	R/W	0x0	
11	M_GEN_CALL	R/W	0x0	
10	M_START_DET	R/W	0x0	
9	M_STOP_DET	R/W	0x0	
8	M_ACTIVITY	R/W	0x0	

7	M_RX_DONE	R/W	0x0	
6	M_TX_ABRT	R/W	0x0	
5	M_RD_REQ	R/W	0x0	
4	M_TX_EMPTY	R/W	0x0	
3	M_TX_OVER	R/W	0x0	
2	M_RX_FULL	R/W	0x0	
1	M_RX_OVER	R/W	0x0	
0	M_RX_UNDER	R/W	0x0	

13.3.2.14 IC_RAW_INTR_STAT

- **Name:** I²C Raw Interrupt Status Register
- **Size:** 32 bits
- **Address offset:** 0x34
- **Read/write access:** read-only

Unlike the IC_INTR_STAT register, these bits are not masked so they always show the true status of the I²C.

31	30	29	28	27	26	25	24
RSVD							
23	22	21	20	19	18	17	16
RSVD							
15	14	13	12	11	10	9	8
DMA_I2C_DONE	MS_CODE_DET	RSVD	ADDR_MATCH	GEN_CALL	START_DET	STOP_DET	ACTIVITY
R	R		R	R	R	R	R
7	6	5	4	3	2	1	0
RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER
R	R	R	R	R	R	R	R

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15	DMA_I2C_DONE	R	0x0	Set when DMA operation is finished.
14	MS_CODE_DET	R	0x0	Indicates whether Master code is indicated in HS mode.
13	RSVD	N/A	-	Reserved
12	ADDR_MATCH	R	0x0	Set when slave address is matched in low power mode
11	GEN_CALL	R	0x0	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling I ² C or when the CPU reads bit 0 of the IC_CLR_GEN_CALL register. I ² C stores the received data in the Rx buffer.
10	START_DET	R	0x0	Indicates whether a START or RESTART condition has occurred on the I ² C interface regardless of whether I ² C is operating in slave or master mode.
9	STOP_DET	R	0x0	Indicates whether a STOP condition has occurred on the I ² C interface regardless of whether I ² C is operating in slave or master mode. Note: There is no status bit for a RESTART condition because it is detected as a normal start condition. The I ² C protocol does not care whether it is a START or RESTART because both conditions start from the IDLE state and send the message to all the slaves on the bus.
8	ACTIVITY	R	0x0	This bit captures I ² C activity and stays set until it is cleared. There are four ways to clear it. <ul style="list-style-type: none"> ● Disabling the I²C ● Reading the IC_CLR_ACTIVITY register ● Reading the IC_CLR_INTR register ● System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the I ² C module is idle, this bit remains set until cleared, indicating that there was activity on the bus.
7	RX_DONE	R	0x0	When the I ² C is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.

6	TX_ABORT	R	0x0	This bit indicates if I ² C, as an I ² C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I ² C master or an I ² C slave. When this bit is set to 1, the IC_TX_ABORT_SOURCE register indicates the reason why the transmit abort takes places.
5	RD_REQ	R	0x0	This bit is set to 1 when I ² C is acting as a slave and another I ² C master is attempting to read data from I ² C. The I ² C holds the I ² C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the IC_DATA_CMD register. This bit is set to 0 just after the processor reads the IC_CLR_RD_REQ register.
4	TX_EMPTY	R	0x0	This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the IC_ENABLE bit 0 is 0, the Tx FIFO is flushed and held in reset. There the Tx FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer activity, then with ic_en=0, this bit is set to 0.
3	TX_OVER	R	0x0	Set during transmit if the transmit buffer is filled to IC_TX_BUFFER_DEPTH and the processor attempts to issue another I ² C command by writing to the IC_DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.
2	RX_FULL	R	0x0	Set when the receive buffer reaches or goes above the RX_TL threshold in the IC_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (IC_ENABLE[0]=0), the Rx FIFO is flushed and held in reset; therefore the Rx FIFO is not full. So this bit is cleared once the IC_ENABLE bit 0 is programmed with a 0, regardless of the activity that continues.
1	RX_OVER	R	0x0	Set if the receive buffer is completely filled to IC_RX_BUFFER_DEPTH and an additional byte is received from an external I ² C device. The I ² C acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.
0	RX_UNDER	R	0x0	Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.

13.3.2.15 IC_RX_TL

- **Name:** I²C Receive FIFO Threshold Register
- **Size:** 32 bits
- **Address offset:** 0x38
- **Read/write access:** read/write

31	30	29	...	10	9	8	7	6	5	...	2	1	0
RSVD										RX_TL			
										R/W			

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7:0	RX_TL	R/W	0x0	Receive FIFO Threshold Level Controls the level of entries (or above) that triggers the <i>RX_FULL</i> interrupt (bit 2 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries.

13.3.2.16 IC_TX_TL

- **Name:** I²C Transmit FIFO Threshold Register
- **Size:** 32 bits
- **Address offset:** 0x3C
- **Read/write access:** read/write

31	30	29	...	10	9	8	7	6	5	...	2	1	0
RSVD								TX_TL					
								R/W					

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7:0	TX_TL	R/W	0x0	Transmit FIFO Threshold Level Controls the level of entries (or below) that triggers the <i>TX_EMPTY</i> interrupt (bit 4 in <i>IC_RAW_INTR_STAT</i> register). The valid range is 0-255, with the additional restriction that it may not be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 0 entry, and a value of 255 sets the threshold for 255 entries.

13.3.2.17 IC_CLR_INTR

- **Name:** Clear Combined and Individual Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0x40
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_INTR
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_INTR	R	0x0	Read this register to clear the combined interrupt, all individual interrupts, and the <i>IC_TX_ABRT_SOURCE</i> register. This bit does not clear hardware clearable interrupts but software clearable interrupts.

13.3.2.18 IC_CLR_RX_UNDER

- **Name:** Clear RX_UNDER Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0x44
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_RX_UNDER
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_RX_UNDER	R	0x0	Read this register to clear the <i>RX_UNDER</i> interrupt (bit 0) of the <i>IC_RAW_INTR_STAT</i> register.

13.3.2.19 IC_CLR_RX_OVER

- **Name:** Clear RX_OVER Interrupt Register
- **Size:** 32 bits

- **Address offset:** 0x48
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_RX_OVER
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_RX_OVER	R	0x0	Read this register to clear the <i>RX_OVER</i> interrupt (bit 1) of the IC_RAW_INTR_STAT register.

13.3.2.20 IC_CLR_TX_OVER

- **Name:** Clear TX_OVER Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0x4C
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_TX_OVER
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_TX_OVER	R	0x0	Read this register to clear the <i>TX_OVER</i> interrupt (bit 3) of the IC_RAW_INTR_STAT register.

13.3.2.21 IC_CLR_RD_REQ

- **Name:** Clear RD_REQ Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0x50
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_RD_REQ
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_RD_REQ	R	0x0	Read this register to clear the <i>RX_REQ</i> interrupt (bit 5) of the IC_RAW_INTR_STAT register.

13.3.2.22 IC_CLR_TX_ABRT

- **Name:** Clear TX_ABRT Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0x54
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_TX_ABRT
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_TX_ABRT	R	0x0	Read this register to clear the <i>TX_ABRT</i> interrupt (bit 6) of the IC_RAW_INTR_STAT register, and the IC_TX_ABRT_SOURCE register.

				The I ² C flushes/resets/empties the Tx FIFO read this register. Once this read is performed, the Tx FIFO is then ready to accept more data bytes from the APB interface.
--	--	--	--	--

13.3.2.23 IC_CLR_RX_DONE

- **Name:** Clear RX_DONE Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0x58
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_RX_DONE
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_RX_DONE	R	0x0	Read this register to clear the <i>RX_DONE</i> interrupt (bit 7) of the IC_RAW_INTR_STAT register.

13.3.2.24 IC_CLR_ACTIVITY

- **Name:** Clear ACTIVITY Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0x5C
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_ACTIVITY
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_ACTIVITY	R	0x0	Reading this register clears the <i>ACTIVITY</i> interrupt if the I ² C is not active anymore. If the I ² C module is still active on the bus, the <i>ACTIVITY</i> interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the <i>ACTIVITY</i> interrupt (bit 8) of the IC_RAW_INTR_STAT register.

13.3.2.25 IC_CLR_STOP_DET

- **Name:** Clear STOP_DET Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0x60
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_STOP_DET
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_STOP_DET	R	0x0	Read this register to clear the <i>STOP_DET</i> interrupt (bit 9) of the IC_RAW_INTR_STAT register.

13.3.2.26 IC_CLR_START_DET

- **Name:** Clear START_DET Interrupt Register
- **Size:** 32 bits

- **Address offset:** 0x64
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_START_DET
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_START_DET	R	0x0	Read this register to clear the <i>START_DET</i> interrupt (bit 10) of the IC_RAW_INTR_STAT register.

13.3.2.27 IC_CLR_GEN_CALL

- **Name:** Clear GEN_CALL Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0x68
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							CLR_GEN_CALL
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	CLR_GEN_CALL	R	0x0	Read this register to clear the <i>GEN_CALL</i> interrupt (bit 11) of the IC_RAW_INTR_STAT register.

13.3.2.28 IC_ENABLE

- **Name:** I²C Enable Register
- **Size:** 32 bits
- **Address offset:** 0x6C
- **Read/write access:** read/write

31	30	29	...	4	3	2	1	0
RSVD							ABORT	ENABLE
							R/W	R/W

Bit	Name	Access	Reset	Description
31:2	RSVD	N/A	-	Reserved
1	ABORT	R/W		Abort I ² C current transfer without flush Tx/Rx FIFO
0	ENABLE	R/W	0x0	Controls whether the I ² C is enabled. <ul style="list-style-type: none"> ● 0: Disables I²C (Tx and Rx FIFOs are held in an erased state) ● 1: Enables I²C Software can disable I ² C while it is active. However, it is important that care be taken to ensure that I ² C is disabled properly. When I ² C is disabled, the following occurs: <ul style="list-style-type: none"> ● The Tx FIFO and Rx FIFO get flushed. ● Status bits in the IC_INTR_STAT register are still active until I²C goes into IDLE state.

13.3.2.29 IC_STATUS

- **Name:** I²C Status Register
- **Size:** 32 bits
- **Address offset:** 0x70
- **Read/write access:** read-only

This is a read-only register used to indicate the current transfer status and FIFO status. The status register may be read at any time. None of the bits in this register request an interrupt.

When the I²C is disabled by writing 0 in bit 0 of the IC_ENABLE register:

- Bits 1 and 2 are set to 1
- Bits 3 and 4 are set to 0
-

When the master or slave state machine goes to idle and ic_en=0:

- Bits 5 and 6 are set to 0

31	30	29	...	9	8	7	6	5	4	3	2	1	0
RSVD							SLV_ACTIVITY	MST_ACTIVITY	RFF	RFNE	TFE	TFNF	ACTIVITY
							R	R	R	R	R	R	R

Bit	Name	Access	Reset	Description
31:7	RSVD	N/A	-	Reserved
6	SLV_ACTIVITY	R	0x0	Slave FSM Activity Status. When the Slave Finite State Machine (FSM) is not in the IDLE state, this bit is set. <ul style="list-style-type: none"> ● 0: Slave FSM is in IDLE state so the Slave part of I²C is not Active ● 1: Slave FSM is not in IDLE state so the Slave part of I²C is Active
5	MST_ACTIVITY	R	0x0	Master FSM Activity Status. When the Master Finite State Machine (FSM) is not in the IDLE state, this bit is set. <ul style="list-style-type: none"> ● 0: Master FSM is in IDLE state so the Master part of I²C is not Active ● 1: Master FSM is not in IDLE state so the Master part of I²C is Active
4	RFF	R	0x0	Receive FIFO Completely Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. <ul style="list-style-type: none"> ● 0: Receive FIFO is not full ● 1: Receive FIFO is full
3	RFNE	R	0x0	Receive FIFO Not Empty. This bit is set when the receive FIFO contains one or more entries; it is cleared when the receive FIFO is empty. <ul style="list-style-type: none"> ● 0: Receive FIFO is empty ● 1: Receive FIFO is not empty
2	TFE	R	0x1	Transmit FIFO Completely Empty. When the transmit FIFO is completely empty, this bit is set. When it contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. <ul style="list-style-type: none"> ● 0: Transmit FIFO is not empty ● 1: Transmit FIFO is empty
1	TFNF	R	0x1	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. <ul style="list-style-type: none"> ● 0: Transmit FIFO is full ● 1: Transmit FIFO is not full
0	ACTIVITY	R	0x0	I ² C Activity Status, is the OR of SLV_ACTIVITY and MST_ACTIVITY bits.

13.3.2.30 IC_TXFLR

- **Name:** I²C Transmit FIFO Level Register
- **Size:** 32 bits
- **Address offset:** 0x74
- **Read/write access:** read-only

This register contains the number of valid data entries in the transmit FIFO buffer. It is cleared whenever:

- The I²C is disabled
- Whenever there is a transmit abort caused by any of the events tracked in the IC_TX_ABRT_SOURCE register and read IC_CLR_TX_ABRT register

The register increments whenever data is placed into the transmit FIFO and decrements when data is taken from the transmit FIFO.

31	30	29	...	8	7	6	5	4	3	2	1	0
----	----	----	-----	---	---	---	---	---	---	---	---	---

RSVD	TXFLR
	R

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	-	Reserved
5:0	TXFLR	R	0x0	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

13.3.2.31 IC_RXFLR

- **Name:** I²C Receive FIFO Level Register
- **Size:** 32 bits
- **Address offset:** 0x78
- **Read/write access:** read

This register contains the number of valid data entries in the receive FIFO buffer. It is cleared whenever:

- The I²C is disabled
- Whenever there is a transmit abort caused by any of the events tracked in the IC_TX_ABRT_SOURCE register and read IC_CLR_TX_ABRT register

The register increments whenever data is placed into the receive FIFO and decrements when data is taken from the receive FIFO.

31	30	29	...	7	6	5	4	3	2	1	0
RSVD							RXFLR				
							R				

Bit	Name	Access	Reset	Description
31:5	RSVD	N/A	-	Reserved
4:0	RXFLR	R	0x0	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO.

13.3.2.32 IC_SDA_HOLD

- **Name:** I²C SDA Hold Time Length Register
- **Size:** 32 bits
- **Address offset:** 0x7C
- **Read/write access:** read/write

This register controls the amount of hold time on the SDA signal after a negative edge of SCL in both master and slave mode, in units of ic_clk period. Writes to this register succeed only when IC_ENABLE=0.

The programmed SDA hold time cannot exceed at any time the duration of the low part of scl. Therefore, the programmed value cannot be larger than N_SCL_LOW-2, where N_SCL_LOW is the duration of the low part of the scl period measured in ic_clk cycles.

31	30	29	...	18	17	16	15	14	13	...	2	1	0
RSVD							IC_SDA_HOLD						
							R/W						

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	IC_SDA_HOLD	R/W	0x1	Sets the required SDA hold time in units of ic_clk period.

13.3.2.33 IC_TX_ABRT_SOURCE

- **Name:** I²C Transmit Abort Source Register
- **Size:** 32 bits
- **Address offset:** 0x80
- **Read/write access:** read-only

31	30	...	16	15	14	13	12	11	10
RSVD					ABRT_SLV_A RBLOST	ABRT_SLVFLUS H_TXFIFO	ARB_LOST	ABRT_MASTER _DIS	ABRT_10B_RD _NORSTRT
					R	R	R	R	R
9	8	7	6	5	4	3	2	1	0
ABRT_SBYTE_ NORSTRT	ABRT_HS_N ORSTRT	ABRT_SBYTE _ACKDET	ABRT_HS_ ACKDET	ABRT_GCAL L_READ	ABRT_GCALL _NOACK	ABRT_TXDATA _NOACK	ABRT_10ADDR 2_NOACK	ABRT_10ADDR 1_NOACK	ABRT_7B_ADD R_NOACK
R	R	R	R	R	R	R	R	R	R

Bit	Name	Access	Reset	Description	Role of I ² C
31:15	RSVD	N/A	-	Reserved	-
14	ABRT_SLV_ARBLOST	R	0x0	1: Slave lost the bus while transmitting data to a remote master. <i>IC_TX_ABRT_SOURCE</i> [12] is set at the same time.	Slave-Transmitter
13	ABRT_SLVFLUSH_TXFIFO	R	0x0	1: Slave has received a read command and some data exists in the Tx FIFO so the slave issues a <i>TX_ABRT</i> interrupt to flush old data in Tx FIFO.	Slave-Transmitter
12	ARB_LOST	R	0x0	1: Master has lost arbitration, or if <i>IC_TX_ABRT_SOURCE</i> [14] is also set, then the slave transmitter has lost arbitration. Note: I ² C can be both master and slave at the same time.	Master-Transmitter or Slave-Transmitter
11	ABRT_MASTER_DIS	R	0x0	1: User tries to initiate a Master operation with the Master mode disabled.	Master-Transmitter or Master-Receiver
10	ABRT_10B_RD_NORSTRT	R	0x0	1: The restart is disabled (<i>IC_RESTART_EN</i> bit (<i>IC_CON</i> [5]) = 0) and the master sends a read command in 10-bit addressing mode.	Master-Receiver
9	ABRT_SBYTE_NORSTRT	R	0x0	1: The restart is disabled (<i>IC_RESTART_EN</i> bit (<i>IC_CON</i> [5]) = 0) and the user is trying to send a START Byte.	Master
8	ABRT_HS_NORSTRT	R	0x0	1: The restart is disabled (<i>IC_RESTART_EN</i> bit (<i>IC_CON</i> [5]) = 0) and the user is trying to use the master to transfer data in High Speed mode.	Master-Transmitter or Master-Receiver
7	ABRT_SBYTE_ACKDET	R	0x0	1: Master has sent a START Byte and the START Byte was acknowledged (wrong behavior).	Master
6	ABRT_HS_ACKDET	R	0x0	1: Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior).	Master
5	ABRT_GCALL_READ	R	0x0	1: I ² C in master mode sent a General Call but the user programmed the byte following the General Call to be a read from the bus (<i>IC_DATA_CMD</i> [9] is set to 1).	Master-Transmitter
4	ABRT_GCALL_NOACK	R	0x0	1: I ² C in master mode sent a General Call and no slave on the bus acknowledged the General Call.	Master-Transmitter
3	ABRT_TXDATA_NOACK	R	0x0	1: This is a master-mode only bit. Master has received an acknowledgement for the address, but when it sent data byte(s) following the address, it did not receive an acknowledge from the remote slave(s).	Master-Transmitter
2	ABRT_10ADDR2_NOACK	R	0x0	1: Master is in 10-bit address mode and the second address byte of the 10-bit address was not acknowledged by any slave.	Master-Transmitter or Master-Receiver
1	ABRT_10ADDR1_NOACK	R	0x0	1: Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave.	Master-Transmitter or Master-Receiver
0	ABRT_7B_ADDR_NOACK	R	0x0	1: Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave.	Master-Transmitter or Master-Receiver

13.3.2.34 IC_SLV_DATA_NACK_ONLY

- **Name:** Generate Slave Data NACK Register
- **Size:** 32 bits
- **Address offset:** 0x84
- **Read/write access:** read/write

A write can occur on this register if either of the following conditions are met:

- I²C is disabled (IC_ENABLE[0] = 0)
- Slave part is inactive (IC_STATUS[6] = 0)

Note: The IC_STATUS[6] is a register read-back location for the internal slv_activity signal; the user should poll this before writing the ic_slv_data_nack_only bit.

31	30	29	...	3	2	1	0
RSVD							NACK
							R/W

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	NACK	R/W	0x0	Generate NACK. This NACK generation only occurs when I ² C is a slave-receiver. If this register is set to a value of 1, it can only generate a NACK after a data byte is received; hence, the data transfer is aborted and the data received is not pushed to the receive buffer. When the register is set to a value of 0, it generates NACK/ACK, depending on normal criteria. <ul style="list-style-type: none"> ● 1: Generate NACK after data byte received ● 0: Generate NACK/ACK normally

13.3.2.35 IC_DMA_CR

- **Name:** DMA Control Register
- **Size:** 32 bits
- **Address offset:** 0x88
- **Read/write access:** read/write

There is a separate bit for transmit and receive. This can be programmed regardless of the state of IC_ENABLE.

31	30	29	...	4	3	2	1	0
RSVD							TDMAE	RDMAE
							R/W	R/W

Bit	Name	Access	Reset	Description
31:2	RSVD	N/A	-	Reserved
1	TDMAE	R/W	0x0	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. <ul style="list-style-type: none"> ● 0: Transmit DMA disabled ● 1: Transmit DMA enabled
0	RDMAE	R/W	0x0	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel. <ul style="list-style-type: none"> ● 0: Receive DMA disabled ● 1: Receive DMA enabled

13.3.2.36 IC_DMA_TDLR

- **Name:** DMA Transmit Data Level Register
- **Size:** 32 bits
- **Address offset:** 0x8C
- **Read/write access:** read/write

31	30	29	...	6	5	4	3	2	1	0
RSVD							DMATDL			
							R/W			

Bit	Name	Access	Reset	Description
31:4	RSVD	N/A	-	Reserved
3:0	DMATDL	R/W	0x0	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is

				generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and $TDMAE = 1$.
--	--	--	--	---

13.3.2.37 IC_DMA_RDLR

- **Name:** I²C Receive Data Level Register
- **Size:** 32 bits
- **Address offset:** 0x90
- **Read/write access:** read/write

31	30	29	...	6	5	4	3	2	1	0
RSVD							DMARDL			
							R/W			

Bit	Name	Access	Reset	Description
31:4	RSVD	N/A	-	Reserved
3:0	DMARDL	R/W	0x0	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL + 1; that is, the dma_rx_req signal is generated when the number of valid data entries in the receive FIFO is equal to or more than this field value + 1, and $RDMAE = 1$. For instance, when DMARDL is 0, then dma_rx_req is asserted when 1 or more data entries are present in the receive FIFO.

13.3.2.38 IC_SDA_SETUP

- **Name:** I²C SDA Setup Register
- **Size:** 32 bits
- **Address offset:** 0x94
- **Read/write access:** read/write

This register controls the amount of time delay (in terms of number of ic_clk clock periods) introduced in the rising edge of SCL—relative to SDA changing—by holding SCL low when I²C services a read request while operating as a slave-transmitter.

The IC_SDA_SETUP register is only used by the I²C when operating as a slave transmitter.

31	30	29	28	...	11	10	9	8	7	6	...	1	0
RSVD										SDA_SETUP			
										R/W			

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7:0	SDA_SETUP	R/W	0x5	SDA Setup. It is recommended that if the required delay is 1000ns, then for an ic_clk frequency of 10MHz, IC_SDA_SETUP should be programmed to a value of 11. IC_SDA_SETUP must be programmed with a minimum value of 2.

13.3.2.39 IC_ACK_GENERAL_CALL

- **Name:** I²C ACK General Call Register
- **Size:** 32 bits
- **Address offset:** 0x98
- **Read/write access:** read/write

The register controls whether I²C responds with a ACK or NACK when it receives an I²C General Call address.

31	30	29	...	3	2	1	0
RSVD							ACK_GEN_CALL
							R/W

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	ACK_GEN_CALL	R/W	0x1	ACK General Call. When set to 1, I ² C responds with a ACK when it receives a General Call. When set to 0, the I ² C does not generate General Call interrupts.

13.3.2.40 IC_ENABLE_STATUS

- **Name:** I²C Enable Status Register
- **Size:** 32 bits
- **Address offset:** 0x9C
- **Read/write access:** read-only

The register is used to report the I²C hardware status when the IC_ENABLE register is set from 1 to 0; that is, when I²C is disabled.

- If IC_ENABLE has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1.
- If IC_ENABLE has been set to 0, bits 2:1 is only be valid as soon as bit 0 is read as '0'.

31	30	29	...	7	6	5	4	3	2	1	0
RSVD							DMA_DISABLE_STATUS	SLV_RX_DATA_LOST	SLV_DISABLE _WHILE_BUSY	IC_EN	
							R	R	R	R	

Bit	Name	Access	Reset	Description
31:5	RSVD	N/A	-	Reserved
4:3	DMA_DISABLE_STATUS	R	0x0	DMA_DISABLE_WHILE_BUSY <ul style="list-style-type: none"> ● 00: No ill disable event is active ● 01: I²C is disable while busy in DMA Legacy mode ● 10: I²C is disable while busy in DMA Register mode
2	SLV_RX_DATA_LOST	R	0x0	Slave Received Data Lost. This bit indicates if a Slave-Receiver operation has been aborted with at least one data byte received from an I ² C transfer due to the setting of IC_ENABLE from 1 to 0.
1	SLV_DISABLED_WHILE_BUSY	R	0x0	Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due to the setting of the IC_ENABLE register from 1 to 0. This bit is set when the CPU writes a 0 to the IC_ENABLE register while: (a) I ² C is receiving the address byte of the Slave-Transmitter operation from a remote master; OR, (b) address and data bytes of the Slave-Receiver operation from a remote master.
0	IC_EN	R	0x0	ic_en Status. This bit always reflects the value driven on the output port ic_en. <ul style="list-style-type: none"> ● When read as 1, I²C is deemed to be in an enabled state. ● When read as 0, I²C is deemed completely inactive.

13.3.2.41 IC_DMA_CMD

- **Name:** I²C DMA Command Register
- **Size:** 32 bits
- **Address offset:** 0xA0
- **Read/write access:** read/write

31	30	...	9	8	7	6	5	4	3	2	1	0
RSVD					DMODE_RESTART	DMODE_STOP	DMODE_CMD	RSVD				SDA_SETUP
					R/W	R/W	R/W					R/W

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7	DMODE_RESTART	R/W	0x0	This bit controls whether a RESTART is issued after the byte is sent or received in DMA mode.

				1: a RESTART is issued after the data is sent/received (according to the value of CMD_RW), regardless of whether or not the transfer direction is changing from the previous command.
6	DMODE_STOP	R/W	0x0	This bit controls whether a STOP is issued after the byte is sent or received in DMA mode. <ul style="list-style-type: none"> 1: STOP is issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus. 0: STOP is not issued after this byte.
5	DMODE_CMD	R/W	0x0	This bit controls whether a read or write is performed in DMA mode. This bit does not control the direction when the I ² C acts as a slave. It controls only the direction when it acts as a master. <ul style="list-style-type: none"> 1: Read 0: Write
4:1	RSVD	N/A	-	Reserved
0	DMODE_ENABLE	R/W	0x0	1: Set to enable DMA mode, clear when transfer is done

13.3.2.42 IC_DMA_DATA_LEN

- **Name:** I²C DMA Mode Transfer Data Length Register
- **Size:** 32 bits
- **Address offset:** 0xA4
- **Read/write access:** read/write

31	30	29	...	18	17	16	15	14	13	...	2	1	0
DMA_DATA_TRD_LEN								DMA_DATA_LEN					
R								R/W					

Bit	Name	Access	Reset	Description
31:16	DMA_DATA_TRD_LEN	R	0x0	DMA mode transfer bytes
15:0	DMA_DATA_LEN	R/W	0x0	DMA transfer data length

13.3.2.43 IC_DMA_MODE

- **Name:** I²C DMA Mode Register
- **Size:** 32 bits
- **Address offset:** 0xA8
- **Read/write access:** read/write

31	30	29	...	4	3	2	1	0
RSVD							DMA_MODE	
							R/W	

Bit	Name	Access	Reset	Description
31:2	RSVD	N/A	-	Reserved
1:0	DMA_MODE	R/W	0x0	DMA Mode Select <ul style="list-style-type: none"> 00: DMA legacy mode 01: DMA Control Register mode

13.3.2.44 IC_SLEEP

- **Name:** I²C Sleep Mode Register
- **Size:** 32 bits
- **Address offset:** 0xAC
- **Read/write access:** read/write

31	30	29	...	4	3	2	1	0
----	----	----	-----	---	---	---	---	---

RSVD		IC_SLEEP_CLK_GATED	IC_SLEEP
		R	R/W

Bit	Name	Access	Reset	Description
31:2	RSVD	N/A	-	Reserved
1	IC_SLEEP_CLK_GATED	R	0x0	I ² C clock has been gated.
0	IC_SLEEP	R/W	0x0	Clock-gated I ² C clock domain for address matching interrupts wake up. <ul style="list-style-type: none"> 1: I²C clock has been gated 0: I²C clock control, write 1 controller would gate I²C clock until I²C slave is enable and reset synchronized register procedure is done.

13.3.2.45 IC_CLR_ADDR_MATCH

- **Name:** Clear Slave Mode Address Match Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0xE4
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							IC_CLR_ADDR_MATCH
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	IC_CLR_ADDR_MATCH	R	0x0	Read this register to clear the slave mode address match interrupt (bit 12) of IC_RAW_INTR_STAT register.

13.3.2.46 IC_CLR_DMA_DONE

- **Name:** Clear DMA_DONE Interrupt Register
- **Size:** 32 bits
- **Address offset:** 0xE8
- **Read/write access:** read-only

31	30	29	...	3	2	1	0
RSVD							IC_CLR_DMA_DONE
							R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	IC_CLR_DMA_DONE	R	0x0	Read this register to clear the DMA_DONE interrupt (bit 15) of IC_RAW_INTR_STAT register.

13.3.2.47 IC_FILTER

- **Name:** I²C Filter Register
- **Size:** 32 bits
- **Address offset:** 0xEC
- **Read/write access:** read/write

31	30	...	10	9	8	7	6	5	4	3	2	1	0
RSVD					IC_DIG_FLTR_SEL	RSVD					IC_DIG_FLTR_DEG		
					R/W						R/W		

Bit	Name	Access	Reset	Description
31:9	RSVD	N/A	-	Reserved

8	IC_DIG_FLTR_SEL	R/W	0x0	1: Enable filter
7:4	RSVD	N/A	-	Reserved
3:0	IC_DIG_FLTR_DEG	R/W	0x0	DIG_FLTR_DEG is to define frequency range of filter. A greater value of DIG_FLTR_DEG results in a slower transfer speed and hardware would be able to filter a lower frequency.

13.3.2.48 IC_COMP_VER

- **Name:** I²C Component Version Register
- **Size:** 32 bits
- **Address offset:** 0xFC
- **Read/write access:** read-only

31	30	29	...	2	1	0
IC_COMP_VERSION						
R						

Bit	Name	Access	Reset	Description
31:0	IC_COMP_VERSION	R	0x20170627	I ² C version number

14 Universal Asynchronous Receiver/Transmitter (UART)

14.1 Introduction

The UART module offers a flexible means of full duplex data exchange with external equipment, requiring an industry standard NRZ asynchronous serial data format. It offers a very wide range of baud rates using a fractional baud rate generator. Low power Rx mode is implemented by monitoring Rx baud rate error and own frequency drift.

14.1.1 Features

- Various UART format: 1 start bit, 7/8 data bits, 0/1 parity bit and 1/2 stop bits
- Very wide range of baud rate
- APB3 bus interface
- Auto-flow control
- Interrupt control
- Infrared data association (IrDA)
- Loop back mode for test
- Separated clocks for Tx path and Rx path
- Fractional baud rate
- Low power mode for Rx path
- Monitor and elimination of Rx baud rate error and own frequency drift automatically for Rx path
- DMA mode
- UART Rx timeout mechanism
- Option for UART Rx to be DMA flow controller

14.1.2 Block Diagram

The block diagram of UART is shown in Fig 14-1.

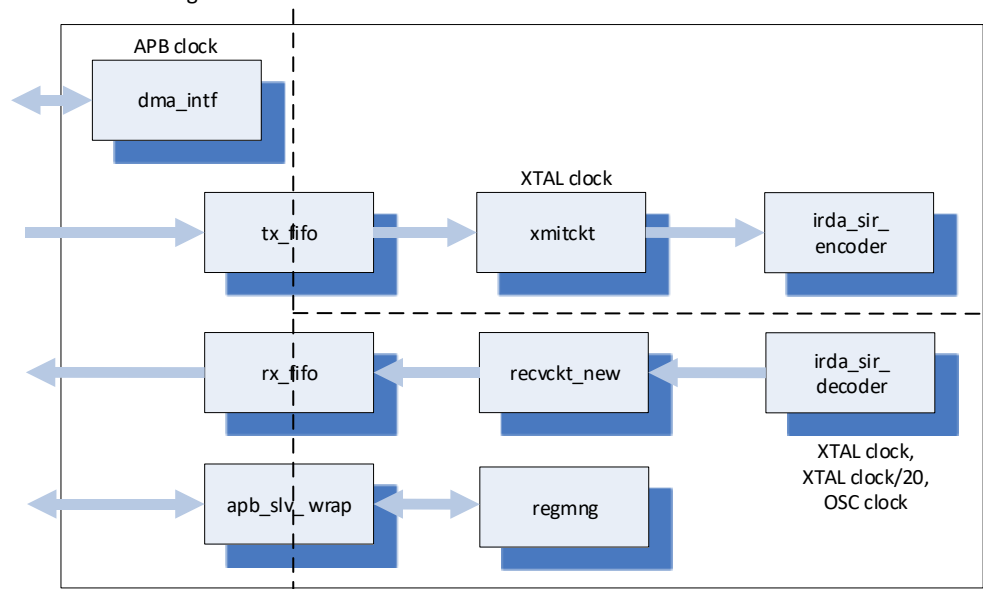


Fig 14-1 UART block diagram

The UART consists of several modules, which provide the following functions:

- dma_intf
 - DMA control unit
 - interface with GDMA module
- tx_fifo
 - asynchronous FIFO
 - 11bit*16
- xmitckt
 - shift register
 - generate UART format data
 - support fractional baud rate
- irda_sir_encoder
 - SIR encoder of UART Tx data
- rx_fifo
 - asynchronous FIFO
 - 11 bits*16
- recvckt_new
 - receive UART format data
 - error detection
 - interrupt control
 - support fractional baud rate
 - oscillator (OSC) clock for low power
 - monitor and eliminate Rx baud rate error and own frequency drift
- irda_sir_decoder
 - SIR decoder of UART Rx data
- apb_slv_wrap
 - APB3 bus interface
- regmng
 - register control unit

14.2 Register

Table 14-1 shows the register map of the UART module. The base address for KM4 UART0 is 0x4000_4000, and KM0 LUART is 0x4800_E000.

Table 14-1 Register map of UART

Name	Address Offset	Access	Description
IER	0x0004	R/W	Enable Interrupt Register
IIR	0x0008	RO	Interrupt Identification Register
LCR	0x000C	R/W	Line Control Register
MCR	0x0010	R/W	Modem Control Register
LSR	0x0014	RO	Line Status Register
MSR	0x0018	RO	Modem Status Register
SCR	0x001C	R/W	Scratch Pad Register
STSR	0x0020	R/W	Factor of Baud Rate Calculation Register
RBR	0x0024	RO	Receiver Buffer Register
THR	0x0024	WO	Transmitter Holding Register
MISCR	0x0028	R/W	DMA Mode and IrDA Mode Control Register
IRDA_SIR_TX_PW_CTRL	0x002C	R/W	IrDA SIR Tx Pulse Width Control Register
IRDA_SIR_RX_PW_CTRL	0x0030	R/W	IrDA SIR Rx Pulse Width Control Register
BAUD_MON	0x0034	R/W	Baud Rate Monitor Register
DBG_UART	0x003C	R/W	Debug Register
REG_RX_PATH_CTRL	0x0040	R/W	Rx Path Control Register
REG_MON_BAUD_CTRL	0x0044	R/W	Baud Rate Monitor Control Register
REG_MON_BAUD_STS	0x0048	R/W	Baud Rate Monitor Status Register
REG_MON_CYC_NUM	0x004C	RO	Clock Cycle Monitored Register It displays the actually monitored clock cycle.

REG_RX_BYTE_CNT	0x0050	R/W	Rx FIFO Byte Count Register It counts the byte number of data reading from Rx FIFO.
FCR	0x0054	R/W	FIFO Control Register

14.2.1 IER

- **Name:** Interrupt Enable Register
- **Size:** 32 bits
- **Address offset:** 0x0004
- **Read/write access:** read/write

This register allows enabling and disabling the interrupt generation by the UART. It can be accessed only when the DLAB bit (LCR[7]) is set to 0.

31	30	29	...		10	9	8
RSVD							
7	6	5	4	3	2	1	0
RSVD		ETOI	EMDI	EDSSI	ELSI	ETBEI	ERBI
		R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	-	Reserved
5	ETOI	R/W	0	Enable Rx Timeout Interrupt <ul style="list-style-type: none"> ● 0: Disabled ● 1: Enabled
4	EMDI	R/W	0	Enable Rx Path Monitor Done Interrupt <ul style="list-style-type: none"> ● 0: Disabled ● 1: Enabled
3	EDSSI	R/W	0	Enable Modem Status Interrupt (EDSSI) (modem status transition) <ul style="list-style-type: none"> ● 0: Disabled ● 1: Enabled
2	ELSI	R/W	0	Enable Receiver Line Status Interrupt (ELSI) (receiver line status) <ul style="list-style-type: none"> ● 0: Disabled ● 1: Enabled
1	ETBEI	R/W	0	Enable Transmitter FIFO Empty Interrupt (ETBEI) (Tx FIFO empty) <ul style="list-style-type: none"> ● 0: Disabled ● 1: Enabled
0	ERBI	R/W	0	Enable Received Data Available Interrupt (ERBFI) (Rx trigger) <ul style="list-style-type: none"> ● 0: Disabled ● 1: Enabled

14.2.2 IIR

- **Name:** Interrupt Identification Register
- **Size:** 32 bits
- **Address offset:** 0x0008
- **Read/write access:** read-only

31	30	...	5	4	3	2	1	0
RSVD					INT_ID[2:0]		INT_PEND	
					R		R	

Bit	Name	Access	Reset	Description
31:4	RSVD	N/A	-	Reserved
3:1	INT_ID[2:0]	R	3'b100	<ul style="list-style-type: none"> ● 3'b011: Interrupt Priority: 1st priority (int_3)

				<ul style="list-style-type: none"> ■ Interrupt Type: Receiver Line Status ■ Interrupt Source: Parity, overrun or framing errors or break interrupt ■ Interrupt Reset Control: Reading the LSR ● 3'b010: <ul style="list-style-type: none"> ■ Interrupt Priority: 2nd priority (int_2) ■ Interrupt Type: Receiver Data Available or trigger level reached ■ Interrupt Source: FIFO Trigger level reached or Rx FIFO full ■ Interrupt Reset Control: FIFO drops below trigger level (depending on FCR[7:6]) ● 3'b110: <ul style="list-style-type: none"> ■ Interrupt Priority: 2nd priority ■ Interrupt Type: Timeout Indication ■ Interrupt Source: There's at least one character in the FIFO but no character has been input to the FIFO or reading from it for the time duration, which depends on the value in register REG_RX_PATH_CTRL[31:16]. ■ Interrupt Reset Control: Reading the RBR or clearing Rx FIFO ● 3'b001: <ul style="list-style-type: none"> ■ Interrupt Priority: 3rd priority ■ Interrupt Type: Tx FIFO empty ■ Interrupt Source: Tx FIFO empty ■ Interrupt Reset Control: Writing to the Tx FIFO (THR) or reading the IIR (if source of interrupt) ● 3'b000: <ul style="list-style-type: none"> ■ Interrupt Priority: 4th priority ■ Interrupt Type: Modem Status ■ Interrupt Source: CTS, DSR, RI, or DCD (input relative signal) ■ Interrupt Reset Control: Reading the MSR ● 3'b100: <ul style="list-style-type: none"> ■ Interrupt Priority: 5th priority (int_4) ■ Interrupt Type: monitor done flag ■ Interrupt Source: Rx path monitor done interrupt ■ Interrupt Reset Control: Reading the BAUD_MON
0	INT_PEND	R	1	<ul style="list-style-type: none"> ● 0: An interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. ● 1: No interrupt is pending.

14.2.3 LCR

- **Name:** Line Control Register
- **Size:** 32 bits
- **Address offset:** 0x000C
- **Read/write access:** read/write

31		30		29		...		10		9		8			
RSVD															
7		6		5		4		3		2		1		0	
DLAB		BREAK_CTRL		STICK_PARITY		EVEN_PARITY_SEL		PARITY_EN		STB		RSVD		WSL0	
R/W		R/W		R/W		R/W		R/W		R/W				R/W	

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7	DLAB	R/W	0	<div>Divisor Latch Access bit</div> <ul style="list-style-type: none"> ● 0: The divisor latches can't be accessed. ● 1: The divisor latches can be accessed. <div>Note:</div> <ul style="list-style-type: none"> ● DLL/DLM only can be access when DLAB = 1. ● IER only can be access when DLAB = 0.

				<ul style="list-style-type: none"> THR/RBR doesn't care the value of DLAB.
6	BREAK_CTRL	R/W	0	Break Control bit <ul style="list-style-type: none"> 0: Break is disabled. 1: The serial out is forced into logic '0' (break state).
5	STICK_PARITY	R/W	0	Stick Parity bit. <ul style="list-style-type: none"> If eps bit is 1, parity bit of character shall be 0. If eps bit is 0, parity bit of character shall be 1.
4	EVEN_PARITY_SEL	R/W	0	Even Parity select <ul style="list-style-type: none"> 0: Odd number of logic '1' is transmitted and checked in each word (data and parity combined). In other words, if the data has an even number of '1' in it, then the parity bit is '1'. 1: Even number of logic '1' is transmitted in each word.
3	PARITY_EN	R/W	0	Parity Enable <ul style="list-style-type: none"> 0: No parity 1: Parity bit is generated on each outgoing character and is checked on each incoming one
2	STB	R/W	0	This bit specifies the number of Stop bits transmitted and received in each serial character. <ul style="list-style-type: none"> 0: 1 stop bit 1: 2 stop bits Note: The receiver always checks the first stop bit only.
1	RSVD	N/A	-	Reserved
0	WLS0	R/W	1	Word length selection <ul style="list-style-type: none"> 0: Data is 7 bits word length. 1: Data is 8 bits word length.

14.2.4 MCR

- Name:** Modem Control Register
- Size:** 32 bits
- Address offset:** 0x0010
- Read/write access:** read/write

31	30	...	7	6	5	4	3	2	1	0
RSVD					AUTOFLOW_EN	LOOPBACK_EN	OUT2	OUT1	RTS	DTR
					R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	-	Reserved
5	AUTOFLOW_EN	R/W	0	Auto Flow Enable (AFE) This bit enables auto flow control.
4	LOOPBACK_EN	R/W	0	Loopback mode <ul style="list-style-type: none"> 0: Normal operation 1: Loopback mode
3	OUT2	R/W	0	This bit controls the output 2 (OUT2_) signal, which is an auxiliary user-designated output. Bit 3 affects the OUT2_ in a manner identical to that described below for bit 0. In loopback mode, connected to Data Carrier Detect (DCD)
2	OUT1	R/W	0	This bit controls the Output 1 (OUT1_) signal, which is an auxiliary user-designated output. Bit 2 affects the OUT1_ in a manner identical to that described below for bit 0. In loopback mode, connected to Ring Indicator (RI) signal input
1	RTS	R/W	0	Request to Send (RTS) signal control <ul style="list-style-type: none"> 0: RTS is logic 1 1: RTS is logic 0 This bit controls the RTS_ output. Bit 1 affects the RTS_ output in a manner identical to that described below for bit 0.
0	DTR	R/W	0	Data Terminal Ready (DTR) signal control

				<ul style="list-style-type: none"> 0: DTR is logic 1 1: DTR is logic 0 <p>This bit controls the DTR_ output. When bit 0 is set to logic 1, the DTR_ output is forced to logic 0; When bit 0 is reset to logic 0, the DTR_ output is forced to logic 1.</p>
--	--	--	--	--

14.2.5 LSR

- Name:** Line Status Register
- Size:** 32 bits
- Address offset:** 0x0014
- Read/write access:** read-only

31	30	29	...	10	9	8	
RSVD							
7	6	5	4	3	2	1	0
RXFIFO_ERR	RSVD	TXFIFO_EMPTY	BREAK_ERR_INT	FRAMING_ERR	PARITY_ERR	OVERRUN_ERR	RXFIFO_DATARDY
R		R	R	R	R	R	

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7	RXFIFO_ERR	R	0	UART Rx FIFO error This bit is set when there is at least one parity error, framing error or break indication in the FIFO. This bit is cleared when the CPU reads the LSR, if there are no subsequent errors in the FIFO.
6	RSVD	N/A	-	Reserved
5	TXFIFO_EMPTY	R	1	Tx FIFO empty indicator
4	BREAK_ERR_INT	R	0	Break Interrupt (BI) indicator <ul style="list-style-type: none"> 0: No break condition in the current character. 1: Sets to logic 1 whenever the received data input is held in the spacing (logic 0) state for a longer than a full word transmission time.
3	FRAMING_ERR	R	0	Framing Error (FE) indicator <ul style="list-style-type: none"> 0: No framing error in the current character. 1: The received character at the top of the FIFO doesn't have a valid stop bit.
2	PARITY_ERR	R	0	Parity Error (PE) indicator <ul style="list-style-type: none"> 0: No parity error in current character. 1: Indicates that the received data character doesn't have the correct even or odd parity, as selected by the even-parity-select bit.
1	OVERRUN_ERR	R	0	Overrun Error (OE) indicator <ul style="list-style-type: none"> 0: No overrun state 1: Indicates that the next character is transferred into the Rx FIFO when the Rx FIFO is full, thereby destroying the previous character.
0	RXFIFO_DATARDY	R	0	Data Ready (DR) indicator <ul style="list-style-type: none"> 0: No characters in the Receiver FIFO. 1: At least one character has been received and transferred into the FIFO.

14.2.6 MSR

- Name:** Modem Status Register
- Size:** 32 bits
- Address offset:** 0x0018
- Read/write access:** read-only

31	30	29	...	10	9	8
RSVD						

7	6	5	4	3	2	1	0
R_DCD	R_RI	R_DSR	R_CTS	D_DCD	TERI	D_DSR	D_CTS
R	R	R	R	R	R	R	R

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7	R_DCD	R	0	Complement of the DCD input or equals to OUT2 (MCR[3]) in loopback mode.
6	R_RI	R	0	Complement of the RI input or equals to OUT1 (MCR[2]) in loopback mode.
5	R_DSR	R	0	Complement of the DSR input or equals to DTR (MCR[0]) in loopback mode.
4	R_CTS	R	1	Complement of the CTS input or equals to RTS (MCR[1]) in loopback mode.
3	D_DCD	R	0	Delta Data Carrier Detect (DDCD) indicator <ul style="list-style-type: none"> 1: The DCD line has changed its state. 0: Otherwise
2	TERI	R	0	Trailing Edge of Ring Indicator (TERI) detector. <ul style="list-style-type: none"> 1: The RI line has changed its state from low to high. 0: Otherwise.
1	D_DSR	R	0	Delta Data Set Ready (DDSR) indicator <ul style="list-style-type: none"> 1: The DSR line has changed its state. 0: Otherwise.
0	D_CTS	R	0	Delta Clear to Send (DCTS) indicator <ul style="list-style-type: none"> 1: The CTS line has changed its state. 0: Otherwise.

14.2.7 SCR

- **Name:** Scratch Pad Register
- **Size:** 32 bits
- **Address offset:** 0x001C
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
RSVD					XFACTOR_ADJ[10:0]		
					R/W		
23	22	21	20	19	18	17	16
XFACTOR_ADJ[10:0]							
R/W							
15	14	13	12	11	10	9	8
RSVD					DBG_SEL[3:0]		
					R/W		
7	6	5	4	3	2	1	0
SCRATCH[7]	SCRATCH[6]	RSVD		PIN_LB_TEST	RSVD		
R/W	R/W			R/W			

Bit	Name	Access	Reset	Description
31:27	RSVD	N/A	-	Reserved
26:16	XFACTOR_ADJ[10:0]	R/W	0	One factor of baud rate calculation for Tx path, that is the ovsvr_adj[10:0] of the baud rate formula.
15:12	RSVD	N/A	-	Reserved
11:8	DBG_SEL[3:0]	R/W	0	Debug port selection
7	SCRATCH[7]	R/W	0	Rx break signal interrupt status, Write 1 to clear.
6	SCRATCH[6]	R/W	0	Rx break signal interrupt enable
5:4	RSVD	N/A	-	Reserved
3	PIN_LB_TEST	R/W	0	For UART IP txd/rxd/rts/cts pin loopback test
2:0	RSVD	N/A	-	Reserved

14.2.8 STSR

- **Name:** Factor of Baud Rate Calculation Register
- **Size:** 32 bits
- **Address offset:** 0x0020
- **Read/write access:** read/write

31	30	...	25	24	23	22	...	5	4	3	2	1	0
RSVD					XFACTOR					RSVD			
					R/W								

Bit	Name	Access	Reset	Description
31:24	RSVD	N/A	-	Reserved
23:4	XFACTOR	R/W	20'd4167	The factor of baud rate calculation, that is the ovsr[19:0] of the baud rate formula, for Tx path and Rx path
3:0	RSVD	N/A	-	Reserved

14.2.9 RBR

- **Name:** Receiver Buffer Register
- **Size:** 32 bits
- **Address offset:** 0x0024
- **Read/write access:** read-only

31	30	29	...		10	9	8
RSVD							
7	6	5	4	3	2	1	0
RXDATABIT7	RXDATABIT6	RXDATABIT5	RXDATABIT4	RXDATABIT3	RXDATABIT2	RXDATABIT1	RXDATABIT0
R	R	R	R	R	R	W	R

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7	RXDATABIT7	R	0	Rx data bit 7
6	RXDATABIT6	R	0	Rx data bit 6
5	RXDATABIT5	R	0	Rx data bit 5
4	RXDATABIT4	R	0	Rx data bit 4
3	RXDATABIT3	R	0	Rx data bit 3
2	RXDATABIT2	R	0	Rx data bit 2
1	RXDATABIT1	R	0	Rx data bit 1
0	RXDATABIT0	R	0	Rx data bit 0
				Note: Bit 0 is the least significant bit. It is the first bit serially received.

14.2.10 THR

- **Name:** Transmitter Holding Register
- **Size:** 32 bits
- **Address offset:** 0x0024
- **Read/write access:** write-only

31	30	29	...		10	9	8
RSVD							
7	6	5	4	3	2	1	0
TXDATABIT7	TXDATABIT6	TXDATABIT5	TXDATABIT4	TXDATABIT3	TXDATABIT2	TXDATABIT1	TXDATABIT0
W	W	W	W	W	W	W	W

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7	TXDATABIT7	W	0	Tx data bit 7
6	TXDATABIT6	W	0	Tx data bit 6
5	TXDATABIT5	W	0	Tx data bit 5
4	TXDATABIT4	W	0	Tx data bit 4
3	TXDATABIT3	W	0	Tx data bit 3
2	TXDATABIT2	W	0	Tx data bit 2
1	TXDATABIT1	W	0	Tx data bit 1
0	TXDATABIT0	W	0	Tx data bit 0

Note: Bit 0 is the least significant bit. It is the first bit serially transmitted.

14.2.11 MISCR

- **Name:** DMA Mode and IrDA Mode Control Register
- **Size:** 32 bits
- **Address offset:** 0x0028
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
RSVD							DUMMY_FLAG
							R/W1C
23	22	21	20	19	18	17	16
DUMMY_DATA							
R/W							
15	14	13	12	11	10	9	8
RXDMA_OWNER	IRDA_RX_INV	IRDA_TX_INV	RXDMA_BURSTSIZE[4:0]				
R/W	R/W	R/W	R/W				
7	6	5	4	3	2	1	0
TXDMA_BURSTSIZE[4:0]					RXDMA_EN	TXDMA_EN	IRDA_ENABLE
R/W					R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:25	RSVD	N/A	-	Reserved
24	DUMMY_FLAG	R/W1C	0	This bit is set when master reads dummy data from UART Rx FIFO, it can be cleared by software by writing the bit.
23:16	DUMMY_DATA	R/W	0	When UART is DMA flow controller and Rx length is multiple of Rx burst size, dummy dma_rx_last is given to GDMA when timeout happens after last burst transfer, dummy data is the read data.
15	RXDMA_OWNER	R/W	0	<ul style="list-style-type: none"> ● 1: UART is DMA flow controller for UART Rx. ● 0: GDMA is DMA flow controller for UART Rx.
14	IRDA_RX_INV	R/W	0	<ul style="list-style-type: none"> ● 1: Invert irda_rx_i. ● 0: Don't invert irda_rx_i.
13	IRDA_TX_INV	R/W	0	<ul style="list-style-type: none"> ● 1: Invert irda_tx_o. ● 0: Don't invert irda_tx_o.
12:8	RXDMA_BURSTSIZE[4:0]	R/W	5'd4	Rx DMA burst size. The maximum value is 16.
7:3	TXDMA_BURSTSIZE[4:0]	R/W	5'd4	Tx DMA burst size. The maximum value is 16.
2	RXDMA_EN	R/W	0	<ul style="list-style-type: none"> ● 1: Rx DMA is enabled (valid when DMA_MODE (FCR[3]) is 1'b1). ● 0: Rx DMA is disabled.
1	TXDMA_EN	R/W	0	<ul style="list-style-type: none"> ● 1: Tx DMA is enabled (valid when DMA_MODE (FCR[3]) is 1'b1). ● 0: Tx DMA is disabled.
0	IRDA_ENABLE	R/W	0	<ul style="list-style-type: none"> ● 1: UART co-works with IrDA SIR mode, which means that txd/rxd are IrDA signals. ● 0: UART mode only.

14.2.12 IRDA_SIR_TX_PW_CTRL

- **Name:** IrDA SIR Tx Pulse Width Control Register
- **Size:** 32 bits
- **Address offset:** 0x002C
- **Read/write access:** read/write

31	30	29	28	27	...	19	18	17	16
UPPERBOUND_SHIFTRIGHT		TXPULSE_UPPERBOUND_SHIFTVAL							
R/W		R/W							
15	14	13	12	11	...	3	2	1	0
LOWBOUND_SHIFTRIGHT		TXPULSE_LOWBOUND_SHIFTVAL							
R/W		R/W							

Bit	Name	Access	Reset	Description
31	UPPERBOUND_SHIFTRIGHT	R/W	0	<ul style="list-style-type: none"> ● 0: Shift left (minus offset value of txplsr[30:16]) ● 1: Shift right (plus offset value of txplsr[30:16])
30:16	TXPULSE_UPPERBOUND_SHIFTVAL	R/W	0	The shift value of SIR Tx pulse's right edge position. The nominal IrDA SIR Tx pulse width is 3/16 bits time. To increase or decrease the right edge position of Tx pulse, change the value.
15	LOWBOUND_SHIFTRIGHT	R/W	0	<ul style="list-style-type: none"> ● 0: Shift left (minus offset value of txplsr[14:0]) ● 1: Shift right (plus offset value txplsr[14:0])
14:0	TXPULSE_LOWBOUND_SHIFTVAL	R/W	0	The shift value of SIR Tx pulse's left edge position. The nominal IrDA SIR Tx pulse width is 3/16 bits time. To increase or decrease the left edge position of Tx pulse, change the value.

14.2.13 IRDA_SIR_RX_PW_CTRL

- **Name:** IrDA SIR Rx Pulse Width Control Register
- **Size:** 32 bits
- **Address offset:** 0x0030
- **Read/write access:** read/write

31	30	29	28	...	19	18	17	16
RSVD								
15	14	13	...	3	2	1	0	
R_SIR_RX_FILTER_THRS							R_SIR_RX_FILTER_TH	
R/W							R/W	

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:1	R_SIR_RX_FILTER_THRS	R/W	0	Threshold of SIR Rx filter. Rx pulse is valid only when Rx pulse width is larger than the threshold.
0	R_SIR_RX_FILTER_EN	R/W	0	Function enable of SIR Rx filter.

14.2.14 BAUD_MON

- **Name:** Baud Rate Monitor Register
- **Size:** 32 bits
- **Address offset:** 0x0034
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
TOGGLE_MON_EN	MON_DATA_VLD	FALLING_THRESH					
R/W	R	R/W					
23	22	21	20	19	18	17	16
MIN_LOW_PERIOD							
R							
15	14	13	12	11	10	9	8
MIN_LOW_PERIOD				MIN_FALL_SPACE			
R				R			
7	6	5	4	3	2	1	0
MIN_FALL_SPACE							
R							

Bit	Name	Access	Reset	Description
31	TOGGLE_MON_EN	R/W	0	Toggle monitor enable
30	MON_DATA_VLD	R	0	Monitor data valid
29:24	FALLING_THRESH	R/W	6'd7	Falling threshold
23:12	MIN_LOW_PERIOD	R	0	Minimum space of low level
11:0	MIN_FALL_SPACE	R	0	Minimum space of adjacent falling edge

14.2.15 DBG_UART

- **Name:** Debug Register
- **Size:** 32 bits
- **Address offset:** 0x003C
- **Read/write access:** read-only

31	30	29	28	...	3	2	1	0
DBG_UART								
R								

Bit	Name	Access	Reset	Description
31:0	DBG_UART	R	32'he40000dc	Debug port output value

14.2.16 REG_RX_PATH_CTRL

- **Name:** Rx Path Control Register
- **Size:** 32 bits
- **Address offset:** 0x0040
- **Read/write access:** read/write

31	30	29	28	27	26	...	21	20	19	18	17	16
RX_TIMEOUT_THRES												
R/W												
15	14	13	12	11	10	...	5	4	3	2	1	0
RSVD		RXBAUD_ADJ[10:0]								R_RST_N	RSVD	
		R/W								R/RX_N		

Bit	Name	Access	Reset	Description
31:16	RX_TIMEOUT_THRES	R/W	16'd64	Rx timeout threshold, unit is one bit time. The corresponding time for this field mustn't be less than 50us.
15:14	RSVD	N/A	-	Reserved
13:3	RXBAUD_ADJ[10:0]	R/W	0	One factor of baud rate calculation for Rx path, similar with XFACTOR_ADJ[10:0].
2	R_RST_NEWRX_N	R/W	1	Reset Rx path, low active
1:0	RSVD	N/A	-	Reserved

14.2.17 REG_MON_BAUD_CTRL

- **Name:** Baud Rate Monitor Control Register
- **Size:** 32 bits
- **Address offset:** 0x0044
- **Read/write access:** read/write

31	30	29	28	27	...	10	9	
RSVD		R_UPD_OSC_IN_XTAL	R_CYCNUM_PERBIT_OSC					
		R/W	R/W					
8	7	6	5	4	3	2	1	0
R_BIT_NUM_THRES								R_MON_BAUD_EN
R/W								R/W

Bit	Name	Access	Reset	Description
31:30	RSVD	N/A	-	Reserved
29	R_UPD_OSC_IN_XTAL	R/W	0	Enable updating parameter R_CYCNUM_PERBIT_OSC when updating parameter R_CYCNUM_PERBIT_XTAL.
28:9	R_CYCNUM_PERBIT_OSC	R/W	0	Average OSC clock cycle number of one bit, for Rx path OSC clock. Software sets the initial value, hardware updates it depending on the monitor result.
8:1	R_BIT_NUM_THRES	R/W	0	Bit number threshold of one monitor period, to get the average clock cycles of one bit, the max value is 127.
0	R_MON_BAUD_EN	R/W	0	Function enable of monitoring Rx baud

14.2.18 REG_MON_BAUD_STS

- **Name:** Baud Rate Monitor Status Register
- **Size:** 32 bits
- **Address offset:** 0x0048
- **Read/write access:** read/write

31	30	29	28	27	...	22	21	20	19	18	...	1	0
RSVD		RO_MON_TOTAL_BIT						RO_MON_RDY	R_CYCNUM_PERBIT_XTAL				
		R						R	R/W				

Bit	Name	Access	Reset	Description
31:29	RSVD	N/A	-	Reserved
28:21	RO_MON_TOTAL_BIT	R	0	Actually monitored bit number
20	RO_MON_RDY	R	0	Indicate that calculation of actual cycle number per bit is finished. It's cleared when R_MON_BAUD_EN is 0.
19:0	R_CYCNUM_PERBIT_XTAL	R/W	0	Average fractional XTAL clock cycle number of one bit, for Rx path XTAL clock. Software sets the initial value, and hardware updates it depending on the monitor result.

14.2.19 REG_MON_CYC_NUM

- **Name:** Clock Cycle Monitored Register
- **Size:** 32 bits
- **Address offset:** 0x004C
- **Read/write access:** read-only

31	30	29	28	27	26	25	...	2	1	0
----	----	----	----	----	----	----	-----	---	---	---

RSVD	RO_MON_TOTAL_CYCLE
	R

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27:0	RO_MON_TOTAL_CYCLE	R	0	Actually monitored clock cycles

14.2.20 REG_RX_BYTE_CNT

- **Name:** Rx FIFO Byte Count Register
- **Size:** 32 bits
- **Address offset:** 0x0050
- **Read/write access:** read/write

31	30	29	...	19	18	17	16
RSVD							CLR_RX_BYTE_CNT
							W1C
15	14	13	12	...	3	2	1
RO_RX_BYTE_CNT							
R							

Bit	Name	Access	Reset	Description
31:17	RSVD	N/A	-	Reserved
16	CLR_RX_BYTE_CNT	W1C	0	Writing 1 to clear RO_RX_BYTE_CNT
15:0	RO_RX_BYTE_CNT	R	1	Counting the byte number of data reading from Rx FIFO

14.2.21 FCR

- **Name:** FIFO Control Register
- **Size:** 32 bits
- **Address offset:** 0x0054
- **Read/write access:** read/write

31	30	29	...	10	9	8
RSVD						
7	6	5	4	3	2	1
RXFIFO_TRIGGER_LEVEL	RSVD			DMA_MODE	CLEAR_TXFIFO	CLEAR_RXFIFO
R/W				R/W	W1C	W1C
						RPT_ERR
						R/W

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7:6	RXFIFO_TRIGGER_LEVEL	R/W	2'b11	Defines the 16 entries Receiver FIFO Interrupt trigger level (0 ~ 15 bytes). <ul style="list-style-type: none"> ● 00: 1 byte ● 01: 4 bytes ● 10: 8 bytes ● 11: 14 bytes
5:4	RSVD	N/A	-	Reserved
3	DMA_MODE	R/W	0	DMA mode enable or disable. <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable
2	CLEAR_TXFIFO	W1C	0	Writing logic '1' to this bit clears the Transmitter FIFO and resets its logic. The shift register isn't cleared, that is transmitting of the current character continues. Writing 1 to this bit is self-clearing.
1	CLEAR_RXFIFO	W1C	0	Writing logic '1' to this bit clears the Receiver FIFO and resets its logic.

				But it doesn't clear the shift register, that is receiving of the current character continues. Writing 1 to this bit is self-clearing.
0	RPT_ERR	R/W	1	Enable or disable the report of Error in RCVR FIFO field in LSR bit [7] <ul style="list-style-type: none"> 0: Disable 1: Enable

14.3 Design Implementation

14.3.1 Baud Rate Calculation

Baud rate calculation of Tx path and Rx path is shown as follows:

$$\text{Average BaudRate} = \frac{\text{clock_freq}}{(\text{ovsr}[19:0] + \frac{\text{ovsr_adj}[0] + \text{ovsr_adj}[1] + \dots + \text{ovsr_adj}[10]}{11})}$$

Tx path and Rx path both support fractional baud rate generator. Parameter ovsr[19:0] and ovsr_adj[i] are configured depending on IP clock frequency and used baud rate.

Total baud rate error shall be less than 3% in order to communicate correctly. Total baud rate error ΔE includes three parts:

$$\Delta E = E1 + E2 + E3$$

where

- E1 is the error of real baud rate of Tx device and expected communicate baud rate.
- E2 is the frequency drift of Rx IP clock.
- E3 is the calculation baud error of Rx device.

For Rx device, E1 and E2 can be both decreased by monitoring baud rate of Rx data, and E3 can be decreased by using fractional baud rate generator.

- If maximum sum of E1 and E2 is small (less than 2%), Rx path with XTAL 40MHz clock can be selected, and it can support higher baud rate.
- If maximum sum of E1 and E2 is large (2%~4%), Rx path with OSC 2MHz clock shall be selected, baud rate that it supports is limited, while it can achieve low power by using OSC clock (±3%).

Table 14-2 gives the supported frequently-used baud rate. For Rx path with 2M clock, E3 shall be less than 3%, so minimum value of ovsr[19:0] is 0.5/3%=16, then maximum supported baud rate is 2MHz/16=125000bps, 0.5 is maximum sample deviation from the middle of one bit.

Table 14-2 Supported baud rate of Tx path & Rx path

Item	Tx/Rx Path (High Speed Mode)	Rx Path (Low Power Mode)
Clock select	40MHz XTAL	XTAL 2MHz/OSC 2MHz(+/-3%)
Supported baud rate (bps)	110, 300, 600, 1200, 9600, 14400, 19200, 28800, 38400, 57600, 76800, 115200, 128000, 153600, 230400, 406800, 500000, 921600, 1000000, 1382400, 1444400, 1500000, 1843200, 2000000, 2100000, 2764800, 3000000, 3250000, 3692300, 3750000, 4000000, 6000000	110, 300, 600, 1200, 9600, 14400, 19200, 28800, 38400, 57600, 76800, 115200

Table 14-3 lists the error rate of different baud rates when IP clock is 40MHz XTAL, the maximum error rate doesn't exceed 0.5%.

Table 14-3 Error rate of baud rate

IP Clock	Theory Baud Rate	Real Baud Rate	Error Rate (%) ⁴
40MHz XTAL	110	110	0
	300	300	0
	600	600	0
	1200	1200	0
	2400	2400	0
	4800	4800	0

⁴ The calculation of error rate retains four decimal places.

9600	9600	0
14400	14400	0
19200	19200	0
28800	28800	0
38400	38399	-0.0026
57600	57604	0.0069
76800	76805	0.0065
115200	115207	0.0061
128000	128000	0
153600	153610	0.0065
230400	230415	0.0065
380400	380228	-0.0452
460800	460829	0.0063
500000	500000	0
921600	921659	0.0064
1000000	1000000	0
1382400	1384083	0.1217
1444400	1444043	-0.0247
1500000	1598127	-0.1249
1843200	1843318	0.0064
2000000	2000000	0
2100000	2105263	0.2506
2764800	2758621	-0.2235
3000000	3007519	0.2506
3250000	3252033	0.0626
3692300	3703704	0.3089
3750000	3738318	-0.3115
4000000	4000000	0
6000000	5970149	-0.4975

14.3.2 Clock Structure of Rx Path

Fig 14-2 to Fig 14-4 gives the clock structure of Rx path for KM0 log UART, KM0 LUART and KM4 UART0.

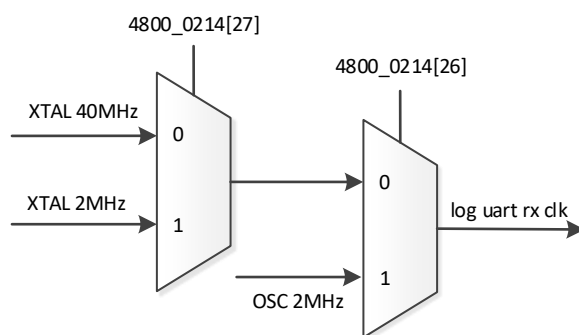


Fig 14-2 Clock structure of KM0 log UART Rx path

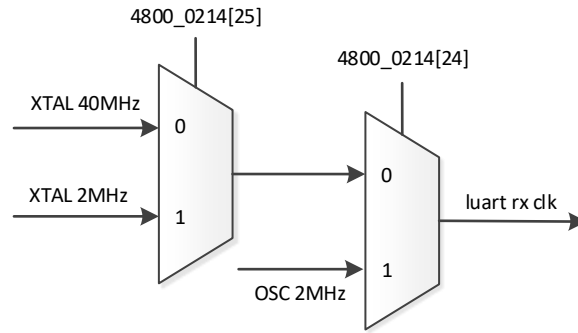


Fig 14-3 Clock structure of KM0 LUART Rx path

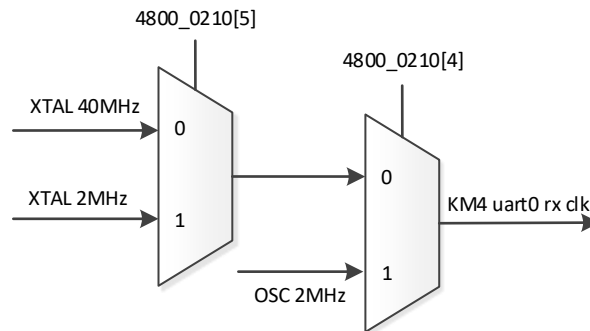


Fig 14-4 Clock structure of KM4 UART0 Rx path

14.3.3 Irda_sir_encoder/decoder

Table 14-4 lists the signaling rate and pulse duration specifications. The UART only supports SIR mode, that is rate up to and including 115.2kbit/s.

Table 14-4 Signaling rate and pulse duration specifications

Signaling Rate	Modulation	Rate Tolerance	Pulse Duration	Pulse Duration	Pulse Duration
2.4kbit/s	RZI	+/-0.87	1.41us	78.13us	88.85us
9.6kbit/s	RZI	+/-0.87	1.41us	19.53us	22.13us
19.2kbit/s	RZI	+/-0.87	1.41us	9.77us	11.07us
38.4kbit/s	RZI	+/-0.87	1.41us	4.88us	5.96us
57.6kbit/s	RZI	+/-0.87	1.41us	3.26us	4.34us
115.2kbit/s	RZI	+/-0.87	1.41us	1.63us	2.23us
0.576Mbit/s	RZI	+/-0.1	295.2ns	434ns	520.8ns
1.152Mbit/s	RZI	+/-0.1	147.6ns	214ns	260.4ns
4.0Mbit/s	4PPM	+/-0.1	115.0ns	125.0ns	135.0ns
4.0Mbit/s	4PPM	+/-0.1	240.0ns	250.0ns	260.0ns
16.0Mbit/s	HHH(1,13)	+/-0.1	38.3ns	41.7ns	45.0ns

Fig 14-5 gives the relationship between IrDA signal and UART signal. Pulse width of IrDA signal is 3-bit or 16-bit width of UART signal. Parameter `ovsr[19:0]` shall be multiple of 16 when IrDA mode is used.

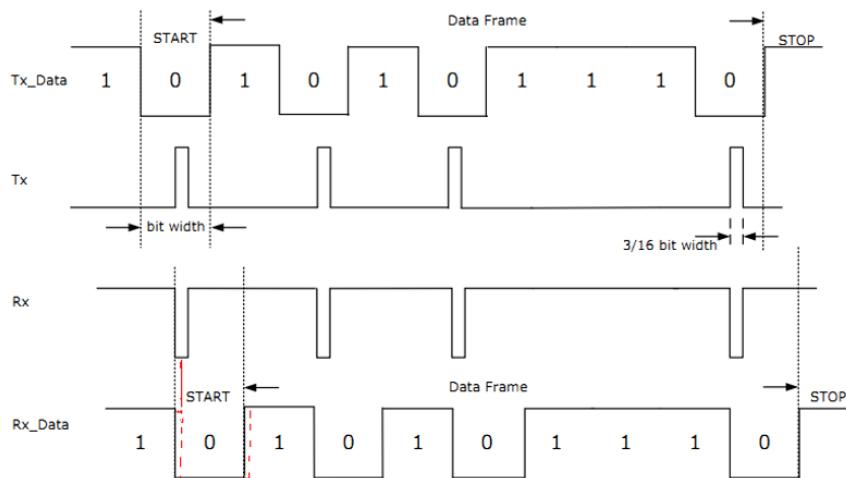


Fig 14-5 Relationship between IrDA signal and UART signal

txd/rxd of sir encoder/decoder is then connected with optical transceiver.

14.3.4 Auto-flow Control

It is possible to control the serial UART data flow between 2 devices by using the nCTS (UCTS_) input and the nRTS (URTS_) output. Fig 14-6 shows how to connect 2 devices in auto-flow control mode.

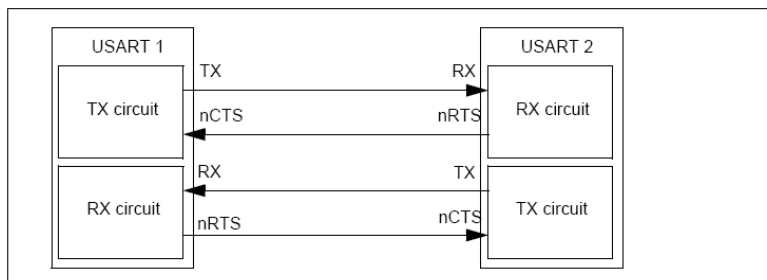


Fig 14-6 Signal connection in auto-flow control mode

When auto-flow control mode is enabled, if input nCTS is deasserted in Tx idle state, the transmission doesn't occur; if input nCTS is deasserted during transmission, the current transmission is completed before the transmitter stops. When Rx FIFO is almost full, nRTS is deasserted to inform peer device to stop transmission at the end of current transmission.

14.3.5 Interrupt Control

Table 14-5 gives the description of different interrupt.

Table 14-5 Description of different interrupt

ID	Priority	Type	Source	Reset Control
3'b011	1st	Receiver Line Status	Parity, overrun or Framing errors or break interrupt	Read the LSR
3'b010	2nd	Receiver Data Available	Rx FIFO Trigger level reached or Rx FIFO full	FIFO drops below trigger level
3'b110	2nd	Timeout Indication	At least one character in the FIFO but no character has been input to the FIFO or read from it for the last 4 characters times	Read the RBR or Clear Rx FIFO
3'b001	3rd	Tx FIFO empty	Tx FIFO empty	Write to the Tx FIFO or read the IIR

3'b000	4th	Modem Status	CTS, DSR, RI, or DCD	Read the MSR
3'b100	5th	Monitor Baud Status	Rx path monitor done interrupt	Read the REG_MON_BAUD_STS

14.3.6 DMA Flow Control

GDMA is always DMA flow controller for UART Tx, while either GDMA or UART can be DMA flow controller for UART Rx. Only Rx DMA mode is illustrated here.

When GDMA is DMA flow controller, GDMA sets expected block length before blocks transfer, and terminates block transfer when Rx length is equal to block length. Rx_dma_req is asserted when Rx size in Rx FIFO isn't less than burst size, and rx_dma_single is asserted when Rx FIFO isn't empty.

When UART is DMA flow controller, UART asserts rx_dma_last to terminate block transfer when UART has received whole packet. The finish flag is defined that no new Rx character comes in for rx_timeout_thres time after the last Rx character.

Fig 14-7 gives the DMA interface timing diagram on condition that block size isn't multiple of burst transaction size. T1 and T2 are burst transfer, T3 and T4 are single transfer. T2 is the last burst transfer of block transfer. After timeout, there are two entries in Rx FIFO which is less than burst transaction transfer size. Then UART starts two single transfers to finish block transfer.

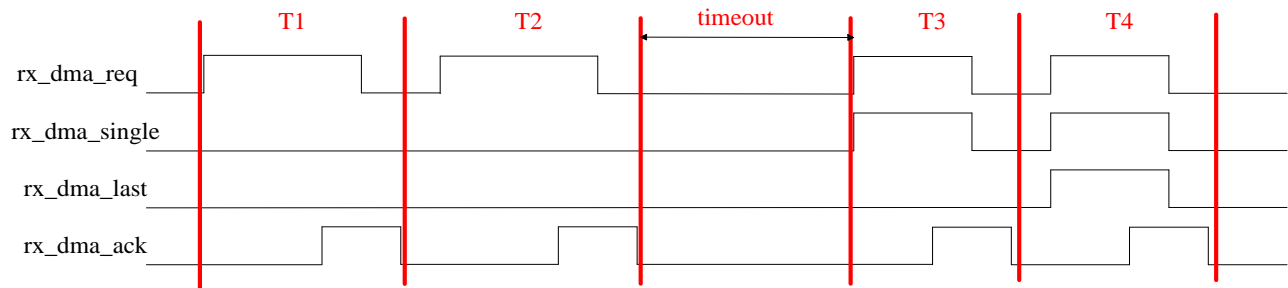


Fig 14-7 DMA interface timing diagram

Fig 14-8 gives the DMA interface timing diagram on condition that block size is multiple of burst transaction size. Block transfer is actually done after last burst transfer of T2. While UART can't judge the end of block transfer until timeout happens. In T3, UART starts a fake single last transfer to inform GDMA, on the same time, UART sets DUMMY_FLAG in the MISCR register for software judgement.

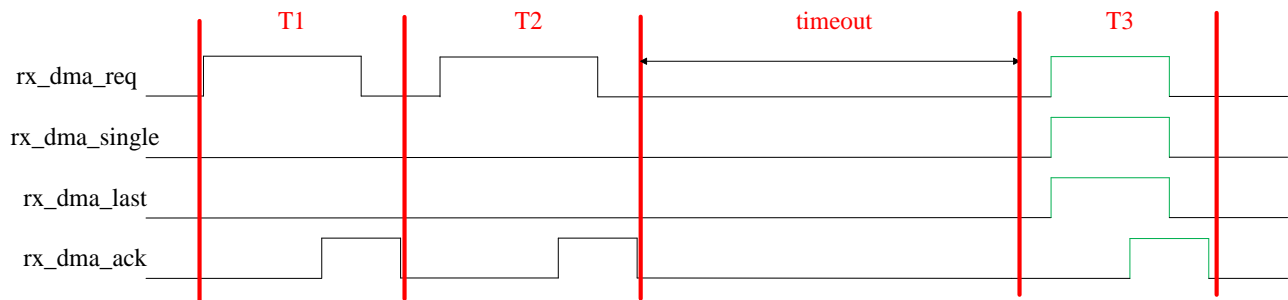


Fig 14-8 DMA interface timing diagram

15 Infrared Radiation (IR)

15.1 Overall Description

15.1.1 Introduction

The IR IP is mainly designed to process IR signal with carrier frequency under 500kHz. The hardware IP supports hardware modulation which can be used on the IR Tx transmission. It also can detect the period of a continuous high or low level signal, and record in Rx FIFO, and then the software can recognize a received IR signal serial and process it.

The IR IP supports 2 types of IR Rx front end, IR diode or IR receiver module. With the IR diode front end, the IR Rx input signal still has carrier clock on it. So the software needs to do the de-modulation, which increases the CPU load. If the IR Rx front end is an IR receiver module, the carrier is filtered by the IR receiver module. So there isn't carrier on the IR Rx input signal.

To simplify the IR signal model, a serial of IR signals is divided into several "symbols". A symbol is defined as a number of carrier clock cycles in a certain period which is called Carrier symbol, or a continuous of high/low level signal for a certain period which is called Space symbol. Carrier symbol is a real signal that spreads in the air; space symbol is the signal before modulation or after de-modulation.

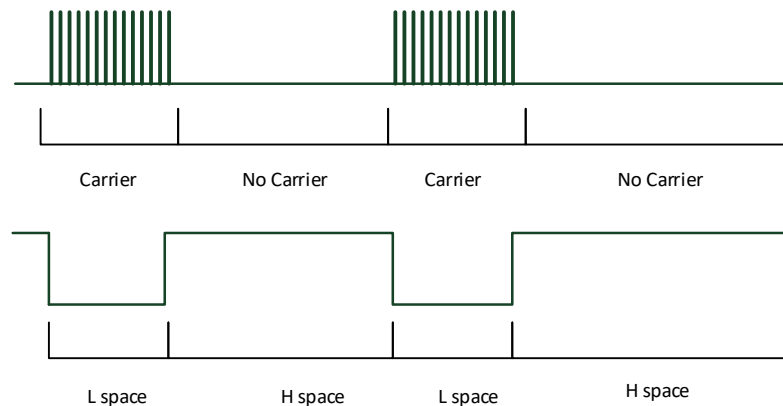


Fig 15-1 IR signal model

In Tx mode, firstly, software programs modulation parameter to generate a fixed frequency carrier, and then programs Tx FIFO to generate space symbols. Tx module modulates and transmits those IR signals. IR Tx flow is shown in Fig 15-2 IR Tx flow.

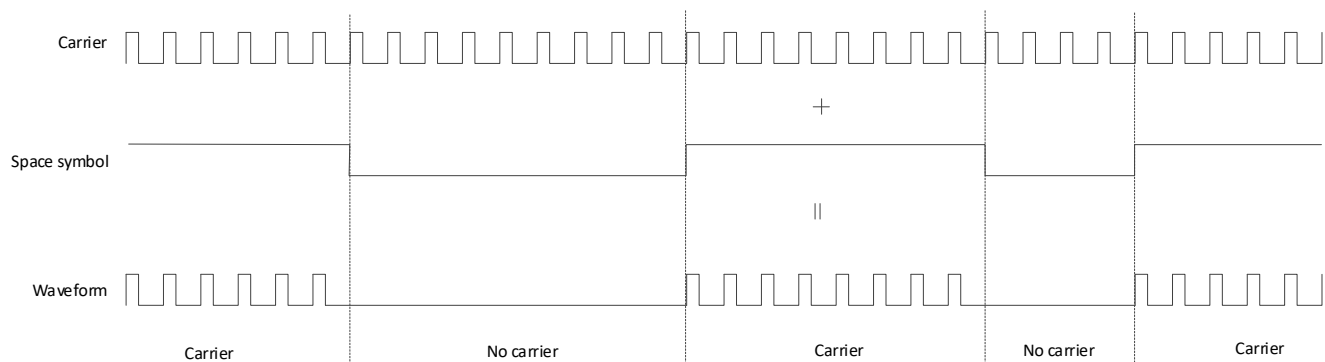


Fig 15-2 IR Tx flow

In Rx mode, if Rx front end is IR receiver module, the signals captured by IR IP core are space symbol; if Rx front end is IR diode, the signals captured by IR IP core are carrier symbol. Software process flow is divided into de-modulation and decoding. If IR diode is used, received IR signals must be de-modulated firstly. IR Rx flow is shown in Fig 15-3.

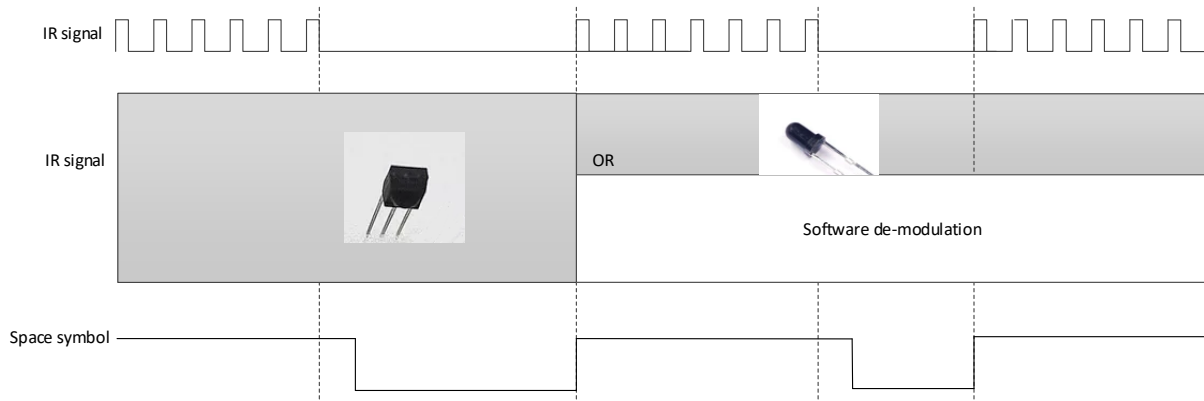


Fig 15-3 IR Rx flow

15.1.2 Features

- Wide range of carrier frequency: from 25kHz to 500kHz
- Customizable duty by users
- Optional IR diode input
- Optional IR receiver module input
- 32*4 bytes Tx FIFO
- 32*4 bytes Rx FIFO
- Configurable Tx carrier frequency
- Configurable Tx carrier duty cycle
- Half duplex mode

15.2 Architecture

The block diagram of IR is shown in Fig 15-4.

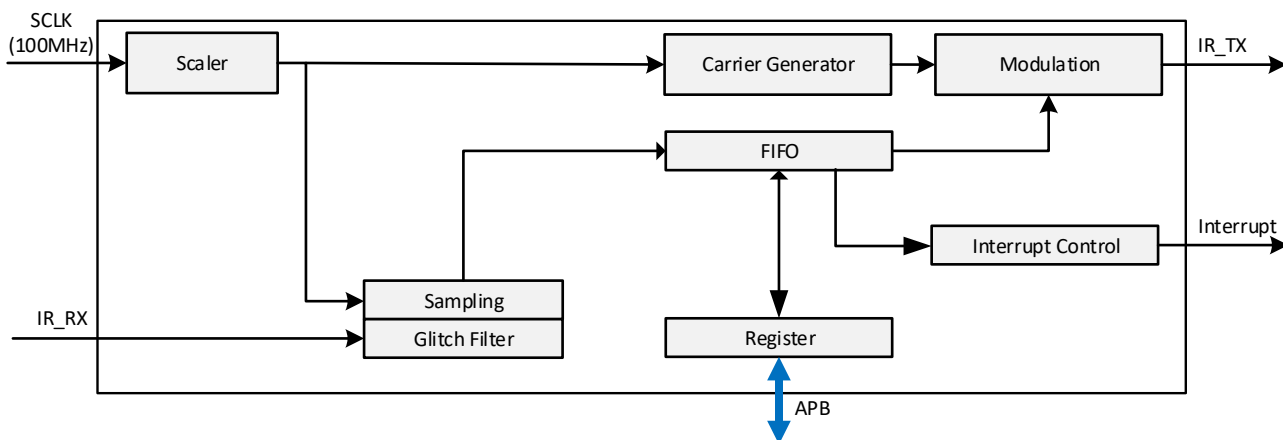


Fig 15-4 IR block diagram

15.2.1 Scaler

- Tx mode: The division clock after scaler is carrier frequency. For example, SCLK=100MHz, carrier frequency = 455kHz, IR_DIV_NUM = $(100M/455k) - 1$.
- Rx mode: The division clock after scaler is sampling clock. Combining Rx counter with sampling clock, software can analyze high or low level signal duration. For example, Rx sampling clock is 10MHz, Rx count is 0x800000FF, it indicates that high level duration is 255us (bit[31] indicates input level 1: high, 0: low).

15.2.2 Glitch Filter

There is a glitch filter before IR input sampling, and the glitch filter can be set to 20ns, 30ns, 40ns, 50ns, 60ns, 70ns, 80ns or 90ns.

15.2.3 Interrupt

Each mode has some interrupt flags (See Table 15-1), and these interrupt sources can be enabled, masked, or cleared individually.

Table 15-1 Interrupts

Mode	Interrupt Flag	Description
Tx mode	IR_TX_FIFO_OVER_INT	When Tx FIFO is full, writing data to Tx FIFO triggers this interrupt.
	IR_TX_FIFO_LEVEL_INT	When Tx FIFO offset decreases from threshold value to threshold value - 1, this interrupt is triggered.
	IR_TX_FIFO_EMPTY_INT	When Tx FIFO offset decreases from 1 to 0, this interrupt is triggered.
Rx mode	IR_RX_FIFO_ERROR_INT	When Rx FIFO is empty, reading data from Rx FIFO triggers this interrupt.
	IR_RX_CNT_THR_INT	When given input level duration exceeds Rx counter threshold, this interrupt is triggered.
	IR_RX_FIFO_OF_INT	When Tx FIFO is full, Rx more data from IR input pin triggers this interrupt.
	IR_RX_FIFO_LEVEL_INT	When Rx FIFO offset increases from threshold value to threshold value + 1, this interrupt is triggered.
	IR_RX_FIFO_FULL_INT	When Rx FIFO is full, this interrupt is triggered.

15.3 Registers

Table 15-2 lists the details of IR registers. The physical base address of IR is 0x4001_2000, and the size is 4KB.

Table 15-2 IR registers

Name	Offset	Access	Description
IR clock control register			
IR_CLK_DIV	0x0000	R/W	IR clock division register
IR Tx registers			
IR_TX_CONFIG	0x0004	R/W	IR Tx configuration register
IR_TX_SR	0x0008	RO	IR Tx FIFO and interrupt status register
IR_TX_COMPE_DIV	0x000C	R/W	IR Tx compensation division register
IR_TX_INT_CLR	0x0010	WC	IR Tx FIFO and interrupt clear register
IR_TX_FIFO	0x0014	W	IR Tx FIFO register
IR Rx registers			
IR_RX_CONFIG	0x0018	R/W	IR Rx configuration register
IR_RX_SR	0x001C	RO	IR Rx FIFO and interrupt status register
IR_RX_INT_CLR	0x0020	WC	IR Rx FIFO and interrupt clear register
IR_RX_CNT_INT_SEL	0x0024	R/W	IR Rx count threshold configure register
IR_RX_FIFO	0x0028	RO	IR Rx FIFO register
IR version register			
IR_VERSION	0x002C	RO	IR IP version register

15.3.1 IR Clock Control Register

15.3.1.1 IR_CLK_DIV

- **Name:** IR clock division register
- **Size:** 32 bits
- **Address offset:** 0x0000
- **Read/write access:** read/write

This register is used for IR Tx carrier clock and Rx sample clock.

31	30	29	28	...	15	14	13	12	11	10	9	...	2	1	0
RSVD									IR_CLK_DIV						
									R/W						

Bit	Name	Access	Reset	Description
31:12	RSVD	N/A	0	Reserved
11:0	IR_CLK_DIV	R/W	0	<p>IR_CLK = IO_CLK/(1 + IR_CLK_DIV)</p> <ul style="list-style-type: none"> ● Tx mode: divider number to generate IrDA modulation frequency. For example: sys_clk = 100MHz, modulation_freq = 455kHz, IR_DIV_NUM = (sys_clk/modulation_freq) - 1 ● Rx mode: waveform sample clock. IR_DIV_NUM = (sys_clk/sample clock) - 1 For example: sample clock = 100MHz, IR_DIV_NUM = 0; sample clock = 50MHz, IR_DIV_NUM = 1

15.3.2 IR Tx Registers

15.3.2.1 IR_TX_CONFIG

- **Name:** IR Tx configuration register
- **Size:** 32 bits
- **Address offset:** 0x0004
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
IR_MODE_SEL	IR_TX_START	RSVD				IR_TX_DUTY_NUM	
R/W	R/W					R/W	
23	22	21	20	19	18	17	16
IR_TX_DUTY_NUM							
R/W							
15	14	13	12	11	10	9	8
RSVD	IR_TX_OUTPUT_INVERSE	IR_TX_DE_INVERSE	IR_TX_FIFO_LEVEL_TH				
	R/W	R/W	R/W				
7	6	5	4	3	2	1	0
RSVD	IR_TX_IDLE_STATE	IR_TX_FIFO_OVERFLOW_INT_MASK	IR_TX_FIFO_OVERFLOW_INT_EN	IR_TX_FIFO_LEVEL_INT_MASK	IR_TX_FIFO_EMPTY_INT_MASK	IR_TX_FIFO_LEVEL_INT_EN	IR_TX_FIFO_EMPTY_INT_EN
	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31	IR_MODE_SEL	R/W	0x0	<ul style="list-style-type: none"> ● 0: Tx mode ● 1: Rx mode
30	IR_TX_START	R/W	0x0	<ul style="list-style-type: none"> ● 0: FSM stops at idle state. ● 1: FSM runs.
29:28	RSVD	N/A	0x0	Reserved

27:16	IR_TX_DUTY_NUM	R/W	0x0	Duty cycle setting for modulation frequency For example: for 1/3 duty cycle, IR_DUTY_NUM = (IR_DIV_NUM+1)/3
15	RSVD	N/A	0x0	Reserved
14	IR_TX_OUTPUT_INVERSE	R/W	0x0	<ul style="list-style-type: none"> 0: Not inverse active output 1: Inverse active output
13	IR_TX_DE_INVERSE	R/W	0x0	<ul style="list-style-type: none"> 0: Not inverse FIFO define 1: Inverse FIFO define
12:8	IR_TX_FIFO_LEVEL_TH	R/W	0x0	Tx FIFO interrupt threshold is from 0 to 15. When Tx FIFO depth = < threshold value, interrupt is triggered.
7	RSVD	N/A	0x0	Reserved
6	IR_TX_IDLE_STATE	R/W	0x0	Tx output state in idle <ul style="list-style-type: none"> 0: Low 1: High
5	IR_TX_FIFO_OVER_INT_MASK	R/W	0x0	Tx FIFO overflow interrupt <ul style="list-style-type: none"> 0: Unmask 1: Mask
4	IR_TX_FIFO_OVER_INT_EN	R/W	0x0	Tx FIFO overflow interrupt <ul style="list-style-type: none"> 0: Disable 1: Enable
3	IR_TX_FIFO_LEVEL_INT_MASK	R/W	0x0	Tx FIFO level interrupt <ul style="list-style-type: none"> 0: Unmask 1: Mask
2	IR_TX_FIFO_EMPTY_INT_MASK	R/W	0x0	Tx FIFO empty interrupt <ul style="list-style-type: none"> 0: Unmask 1: Mask
1	IR_TX_FIFO_LEVEL_INT_EN	R/W	0x0	Tx FIFO level interrupt When Tx FIFO offset = < threshold value, interrupt is triggered. <ul style="list-style-type: none"> 0: Disable 1: Enable
0	IR_TX_FIFO_EMPTY_INT_EN	R/W	0x0	Tx FIFO empty interrupt <ul style="list-style-type: none"> 0: Disable 1: Enable

15.3.2.2 IR_TX_SR

- **Name:** IR Tx interrupt status register
- **Size:** 32 bits
- **Address offset:** 0x0008
- **Read/write access:** read

31	30	29	...	18	17	16
RSVD						
15	14	13	12	11	10	9
IR_TX_FIFO_EMPTY	IR_TX_FIFO_FULL	IR_TX_FIFO_OFFSET				
R	R	R				
7	6	5	4	3	2	1
RSVD			IR_TX_STATUS	RSVD	IR_TX_FIFO_OVE R_INT_STATUS	IR_TX_FIFO_LEVE L_INT_STATUS
			R		R	IR_TX_FIFO_EMP TY_INT_STATUS
					R	R

Bit	Name	Access	Reset	Description
31:16	RSVD	-	0	Reserved
15	IR_TX_FIFO_EMPTY	R	0	<ul style="list-style-type: none"> 0: Not empty 1: Empty
14	IR_TX_FIFO_FULL	R	0	<ul style="list-style-type: none"> 0: Not full 1: Full
13:8	IR_TX_FIFO_OFFSET	R	0	Tx FIFO offset is from 0 to 32.

				Note: After Tx last packet, hardware can't clear Tx FIFO offset.
7:5	RSVD	N/A	0	Reserved
4	IR_TX_STATUS	R	0	<ul style="list-style-type: none"> 0: Idle 1: Run
3	RSVD	N/A	0	Reserved
2	IR_TX_FIFO_OVERFLOW_INT_STATUS	R	0	Tx FIFO overflow interrupt <ul style="list-style-type: none"> 0: Interrupt inactive 1: Interrupt active
1	IR_TX_FIFO_LEVEL_INT_STATUS	R	0	When Tx FIFO offset = < threshold value, interrupt is triggered. <ul style="list-style-type: none"> 0: Interrupt inactive 1: Interrupt active
0	IR_TX_FIFO_EMPTY_INT_STATUS	R	0	Tx FIFO empty interrupt <ul style="list-style-type: none"> 0: Interrupt inactive 1: Interrupt active

15.3.2.3 IR_TX_COMPE_DIV

- Name:** IR Tx compensation division register
- Size:** 32 bits
- Address offset:** 0x000C
- Read/write access:** read/write

31	30	29	28	...	15	14	13	12	11	10	9	...	2	1	0
RSVD										TX_COMPE_DIV					
										R/W					

Bit	Name	Access	Reset	Description
31:12	RSVD	N/A	0	Reserved
11:0	TX_COMPE_DIV	R/W	0	$IR_TX_CLK_Period = SCLK / (TX_COMPE_DIV + 1)$

15.3.2.4 IR_TX_INT_CLR

- Name:** IR Tx interrupt clear register
- Size:** 32 bits
- Address offset:** 0x0010
- Read/write access:** write

31	30	29	...	6	5	4	3	2	1	0
RSVD							IR_TX_FIFO_OVERFLOW_INT_CLR	IR_TX_FIFO_LEVEL_INT_CLR	IR_TX_FIFO_EMPTY_INT_CLR	IR_TX_FIFO_CLR
							WC	WC	WC	WC

Bit	Name	Access	Reset	Description
31:4	RSVD	N/A	0	Reserved
3	IR_TX_FIFO_OVERFLOW_INT_CLR	WC	-	Tx FIFO overflow interrupt Write 1 to clear
2	IR_TX_FIFO_LEVEL_INT_CLR	WC	-	When Tx FIFO offset = < threshold value, interrupt is triggered. Write 1 to clear
1	IR_TX_FIFO_EMPTY_INT_CLR	WC	-	Tx FIFO empty interrupt Write 1 to clear
0	IR_TX_FIFO_CLR	WC	-	Write 1 to clear Tx FIFO

15.3.2.5 IR_TX_FIFO

- Name:** IR Tx FIFO register

- **Size:** 32 bits
- **Address offset:** 0x0014
- **Read/write access:** write

31	30	29	28	27	26	25	...	2	1	0
IR_TX_DATA_TYPE	IR_TX_DATA_END_FLAG	IR_TX_COMPENSATION		IR_TX_DATA_TIME						
W	W	W		W						

Bit	Name	Access	Reset	Description
31	IR_TX_DATA_TYPE	W	0	Data type <ul style="list-style-type: none"> ● 00: Inactive carrier (no carrier) ● 01: Active carrier (carrier)
30	IR_TX_DATA_END_FLAG	W	0	<ul style="list-style-type: none"> ● 0: Normal packet ● 1: Last packet
29:28	IR_TX_COMPENSATION	W	0	<ul style="list-style-type: none"> ● 0x0: IR_TX_CLK_Period = Tsys_clk * IR_CLK_DIV ● 0x1: IR_TX_CLK_Period = (1 + 1/2) Tsys_clk * IR_CLK_DIV ● 0x2: IR_TX_CLK_Period = (1 + 1/4) Tsys_clk * IR_CLK_DIV ● 0x3: IR_TX_CLK_Period = Tsys_clk * (IR_TX_COMPE_DIV + 1)
27:0	IR_TX_DATA_TIME	W	0	Real active time = (IR_TX_DATA_TIME + 1) * IR_TX_CLK_Period

15.3.3 IR Rx Registers

15.3.3.1 IR_RX_CONFIG

- **Name:** IR Rx configuration register
- **Size:** 32 bits
- **Address offset:** 0x0018
- **Read/write access:** read/write

31		30		29		28		27		26		25		24			
RSVD						IR_RX_START		IR_RX_START_M ODE		IR_RX_MAN_STA RT		IR_RX_TRIGGER_MODE					
						R/W		R/W		R/W		R/W					
23				22		21		20		19		18		17		16	
IR_RX_FILTER_STAGETX						RSVD		IR_RX_FIFO_ERR OR_INT_MASK		IR_RX_CNT_THR_ INT_MASK		IR_RX_FIFO_OF_I NT_MASK		IR_RX_CNT_OF_I NT_MASK			
R/W								R/W		R/W		R/W		R/W			
15				14		13		12		11		10		9		8	
IR_RX_FIFO_LEVE L_INT_MASK		IR_RX_FIFO_FULL _INT_MASK		IR_RX_FIFO_DISC ARD_SET		IR_RX_FIFO_LEVEL_TH											
R/W		R/W		R/W		R/W											
7				6		5		4		3		2		1		0	
RSVD				IR_RX_FIFO_ERR OR_INT_EN		IR_RX_CNT_THR_ INT_EN		IR_RX_FIFO_OF_I NT_EN		IR_RX_CNT_OF_I NT_EN		IR_RX_FIFO_LEVE L_INT_EN		IR_RX_FIFO_FULL _INT_EN			
				R/W		R/W		R/W		R/W		R/W		R/W			

Bit	Name	Access	Reset	Description
31:29	RSVD	N/A	0	Reserved
28	IR_RX_START	R/W	0x0	<ul style="list-style-type: none"> ● 0: FSM stops at idle state. ● 1: FSM runs.
27	IR_RX_START_MODE	R/W	0x0	<ul style="list-style-type: none"> ● 0: Manual mode, IR_RX_MAN_START control ● 1: Auto-mode, Trigger-mode control
26	IR_RX_MAN_START	R/W	0x0	If IR_RX_TRIGGER_MODE = 0, writing 1 means starting to check the waveform.
25:24	IR_RX_TRIGGER_MODE	R/W	0x0	<ul style="list-style-type: none"> ● 00: High -> low trigger ● 01: Low -> high trigger ● 10: High -> low or low -> high trigger

23:21	IR_RX_FILTER_STAGETX	R/W	0x0	<ul style="list-style-type: none"> ● 0x0: Filter <= 20ns glitch ● 0x1: Filter <= 30ns glitch ● 0x2: Filter <= 40ns glitch ● 0x3: Filter <= 50ns glitch ● ...
20	RSVD	N/A	0	Reserved
19	IR_RX_FIFO_ERROR_INT_MASK	R/W	0x0	Rx FIFO error read interrupt When Rx FIFO is empty, read Rx FIFO and trigger interrupt. <ul style="list-style-type: none"> ● 0: Unmask ● 1: Mask
18	IR_RX_CNT_THR_INT_MASK	R/W	0x0	Rx count threshold interrupt <ul style="list-style-type: none"> ● 0: Unmask ● 1: Mask
17	IR_RX_FIFO_OF_INT_MASK	R/W	0x0	Rx FIFO overflow <ul style="list-style-type: none"> ● 0: Unmask ● 1: Mask
16	IR_RX_CNT_OF_INT_MASK	R/W	0x0	RX counter overflow <ul style="list-style-type: none"> ● 0: Unmask ● 1: Mask
15	IR_RX_FIFO_LEVEL_INT_MASK	R/W	0x0	Rx FIFO level interrupt <ul style="list-style-type: none"> ● 0: Unmask ● 1: Mask When Rx FIFO offset \geq threshold value, this interrupt is triggered.
14	IR_RX_FIFO_FULL_INT_MASK	R/W	0x0	Rx FIFO full interrupt <ul style="list-style-type: none"> ● 0: Unmask ● 1: Mask
13	IR_RX_FIFO_DISCARD_SET	R/W	0x0	When FIFO is full, new data is send to FIFO: <ul style="list-style-type: none"> ● 0: Discard oldest data in FIFO. ● 1: Reject new data sending to FIFO.
12:8	IR_RX_FIFO_LEVEL_TH	R/W	0x0	Rx FIFO interrupt threshold When Rx FIFO depth > threshold value, this interrupt is triggered.
7:6	RSVD	N/A	0	Reserved
5	IR_RX_FIFO_ERROR_INT_EN	R/W	0x0	Rx FIFO error read interrupt When Rx FIFO is empty, reading the Rx FIFO triggers this interrupt. <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable
4	IR_RX_CNT_THR_INT_EN	R/W	0x0	Rx count threshold interrupt <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable
3	IR_RX_FIFO_OF_INT_EN	R/W	0x0	RX FIFO overflow <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable
2	IR_RX_CNT_OF_INT_EN	R/W	0x0	RX counter overflow <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable
1	IR_RX_FIFO_LEVEL_INT_EN	R/W	0x0	Rx FIFO level interrupt When Rx FIFO offset \geq threshold value, this interrupt is triggered. <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable
0	IR_RX_FIFO_FULL_INT_EN	R/W	0x0	Rx FIFO full interrupt <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable

15.3.3.2 IR_RX_SR

- **Name:** IR clock division register

- **Size:** 32 bits
- **Address offset:** 0x001C
- **Read/write access:** read

31	30	29	28	27	26	25	24
RSVD							
23	22	21	20	19	18	17	16
RSVD						IR_RX_FIFO_EMPTY	IR_RX_FIFO_FULL
15	14	13	12	11	10	9	8
RSVD		IR_RX_FIFO_OFFSET					
		R					
7	6	5	4	3	2	1	0
IR_RX_STATE	RSVD	IR_RX_FIFO_ERROR_INT_STATUS	IR_RX_CNT_THR_INT_STATUS	IR_RX_FIFO_OF_INT_STATUS	IR_RX_CNT_OF_INT_STATUS	IR_RX_FIFO_LEVEL_INT_STATUS	IR_RX_FIFO_FULL_INT_STATUS
R		R	R	R	R	R	R

Bit	Name	Access	Reset	Description
31:18	RSVD	N/A	0	Reserved
17	IR_RX_FIFO_EMPTY	R	0x0	<ul style="list-style-type: none"> ● 0: Not empty ● 1: Empty
16	IR_RX_FIFO_FULL	R	0x0	<ul style="list-style-type: none"> ● 0: Not full ● 1: Full
15:14	RSVD	N/A	0	Reserved
13:8	IR_RX_FIFO_OFFSET	R	0x0	Rx FIFO offset
7	IR_RX_STATE	R	0x0	<ul style="list-style-type: none"> ● 0: Idle ● 1: Run
6	RSVD	N/A	0	Reserved
5	IR_RX_FIFO_ERROR_INT_STATUS	R	0x0	Rx FIFO error read interrupt status When Rx FIFO is empty, reading the Rx FIFO triggers this interrupt. <ul style="list-style-type: none"> ● 0: Interrupt is inactive ● 1: Interrupt is active
4	IR_RX_CNT_THR_INT_STATUS	R	0x0	Rx count threshold interrupt status <ul style="list-style-type: none"> ● 0: Interrupt is inactive ● 1: Interrupt is active
3	IR_RX_FIFO_OF_INT_STATUS	R	0x0	Rx FIFO overflow interrupt status <ul style="list-style-type: none"> ● 0: Interrupt is inactive ● 1: Interrupt is active
2	IR_RX_CNT_OF_INT_STATUS	R	0x0	Rx counter overflow interrupt status <ul style="list-style-type: none"> ● 0: Interrupt is inactive ● 1: Interrupt is active
1	IR_RX_FIFO_LEVEL_INT_STATUS	R	0x0	Rx FIFO level interrupt status <ul style="list-style-type: none"> ● 0: Interrupt is inactive ● 1: Interrupt is active
0	IR_RX_FIFO_FULL_INT_STATUS	R	0x0	Rx FIFO full interrupt status <ul style="list-style-type: none"> ● 0: Interrupt is inactive ● 1: Interrupt is active

15.3.3.3 IR_RX_INT_CLR

- **Name:** IR clock division register
- **Size:** 32 bits
- **Address offset:** 0x0020
- **Read/write access:** write

31	30	29	...	11	10	9	8
RSVD							IR_RX_FIFO_CLR
							WC

7	6	5	4	3	2	1	0
RSVD		IR_RX_FIFO_ERROR	IR_RX_CNT_THR	IR_RX_FIFO_OF_I	IR_RX_CNT_OF_I	IR_RX_FIFO_LEV	IR_RX_FIFO_FUL
		_INT_CLR	_INT_CLR	_NT_CLR	_NT_CLR	EL_INT_CLR	L_INT_CLR
		WC	WC	WC	WC	WC	WC

Bit	Name	Access	Reset	Description
31:9	RSVD	N/A	0	Reserved
8	IR_RX_FIFO_CLR	WC	-	Write 1 to clear Rx FIFO
7:6	RSVD	N/A	0	Reserved
5	IR_RX_FIFO_ERROR_INT_CLR	WC	-	Rx FIFO error read interrupt Write 1 to clear
4	IR_RX_CNT_THR_INT_CLR	WC	-	Rx count threshold interrupt Write 1 to clear
3	IR_RX_FIFO_OF_INT_CLR	WC	-	Rx FIFO overflow interrupt Write 1 to clear
2	IR_RX_CNT_OF_INT_CLR	WC	-	Rx counter overflow interrupt Write 1 to clear
1	IR_RX_FIFO_LEVEL_INT_CLR	WC	-	Rx FIFO level interrupt Write 1 to clear
0	IR_RX_FIFO_FULL_INT_CLR	WC	-	Rx FIFO full interrupt Write 1 to clear

15.3.3.4 IR_RX_CNT_INT_SEL

- **Name:** IR clock division register
- **Size:** 32 bits
- **Address offset:** 0x0024
- **Read/write access:** read/write

31	30	29	28	27	...	3	2	1	0
IR_RX_CNT_THR_TRIGGER_LV	IR_RX_CNT_THR								
R/W	R/W								

Bit	Name	Access	Reset	Description
31	IR_RX_CNT_THR_TRIGGER_LV	R/W	0x0	Trigger level <ul style="list-style-type: none"> ● 0: When low level counter \geq threshold, trigger interrupt ● 1: When high level counter \geq threshold, trigger interrupt
30:0	IR_RX_CNT_THR	R/W	0x0	31-bits threshold

15.3.3.5 IR_RX_FIFO

- **Name:** IR clock division register
- **Size:** 32 bits
- **Address offset:** 0x0028
- **Read/write access:** read

Note: User can't read this register under Tx mode.

31	30	29	28	27	26	...	4	3	2	1	0
IR_RX_LEVEL	IR_RX_CNT										
RO	RO										

Bit	Name	Access	Reset	Description
31	IR_RX_LEVEL	RO	0x0	Rx Level <ul style="list-style-type: none"> ● 1: High level ● 0: Low level

30:0	IR_RX_CNT	RO	0x0	31-bits cycle duration
------	-----------	----	-----	------------------------

15.3.4 IR Version Register

15.3.4.1 IR_VERSION

- **Name:** IR IP version register
- **Size:** 32 bits
- **Address offset:** 0x002C
- **Read/write access:** read

31	30	29	28	27	26	25	26	...	7	6	5	4	3	2	1	0
IR_VERSION																
RO																

Bit	Name	Access	Reset	Description
31:0	IR_VERSION	RO	1410150A	IR IP version

15.4 IR Application Note

There are two typical application scenarios, RCU application and receiver application.

- For RCU application, except for sending IR signals, Ameba-D also supports IR learning function.
- For receiver application, Ameba-D supports IR diode input and module input.

15.4.1 RCU Application

15.4.1.1 IR Tx

- (1) Enable IR clock, set RCC configuration register.
- (2) Disable all interrupts.
- (3) Set IR_CLK_DIV and IR_TX_DUTY_NUM.
- (4) Set IR Tx output level characteristic, IR_TX_OUTPUT_INVERSE/IR_TX_DE_INVERSE/IR_TX_IDEL_STATE (See Fig 15-5).
- (5) Set Tx FIFO threshold if needed.
- (6) Clear all interrupts.
- (7) Set IR work mode.
- (8) Write data to Tx FIFO.
- (9) Start transfer by writing 1 to IR_TX_START.
- (10) Write more data to Tx FIFO if needed (based on TX_FIFO_LEVEL_INT).

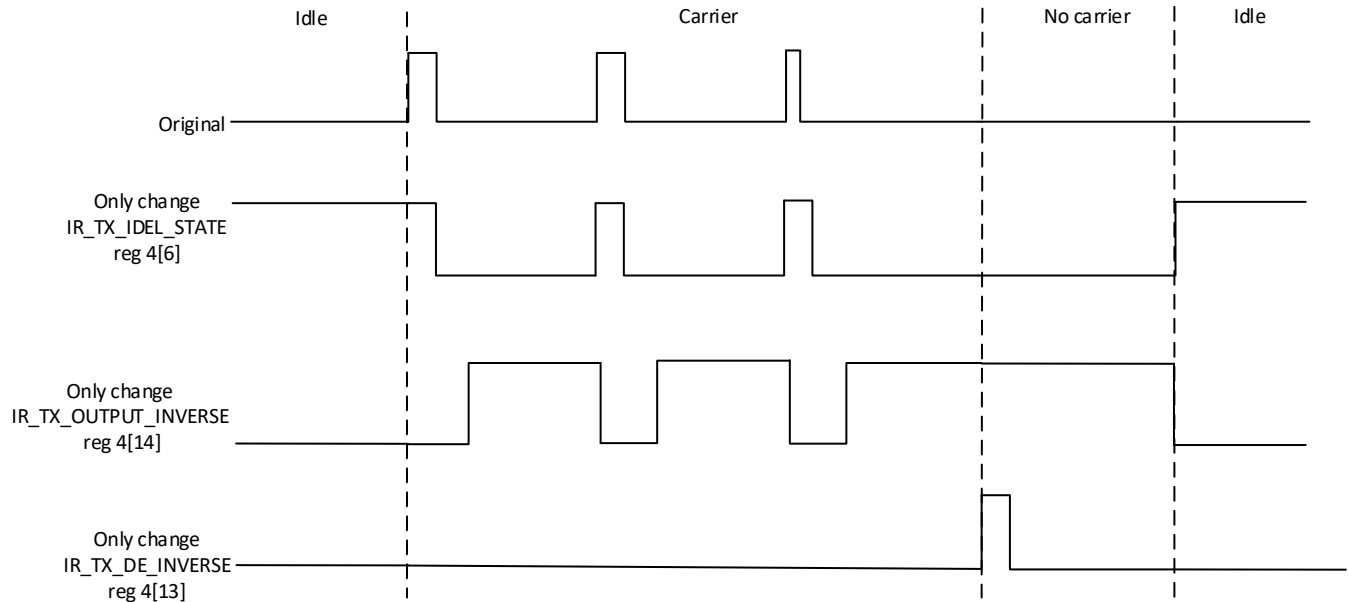


Fig 15-5 Tx output level

15.4.1.2 IR Learning

- (1) Enable IR clock, set RCC configure register.
- (2) Disable all interrupt.
- (3) Set sample clock use IR_CLK_DIV.
- (4) Set IR Rx work in auto mode and configure IR_RX_TRIGGER_MODE.
- (5) Set Rx FIFO threshold if needed.
- (6) Clear all interrupt.
- (7) Set IR work mode.
- (8) Set RX_TRIGGER_MODE
- (9) Start IR Rx FSM.
- (10) Read Data from Rx FIFO

15.4.2 Receiver Application

For receiver application, initialization process is the same as IR learning. It is important to note that diode input increases CPU load, because software must de-module carrier signal.

16 Key-Scan

16.1 Overall Description

16.1.1 Application Scenario

As a keypad scan device, Key-Scan can be applied to simple key, remote control or even game pad. It needs to scan the operations of key press and release accurately and timely.

The major benefit of this device is to free up the CPU from scanning the keypad all the time. It triggers the corresponding interrupts to inform CPU in time. In addition, chip can enter low-power state in most of time, and take little time to wake up and handle the key events.

16.1.2 Features

- Up to 6 * 6 (36) keypad array with use of 12 GPIOs
- Configurable rows and columns of keypad array
- Hardware debounce with configurable time at each scan
- Configurable Scan Clock, Scan Interval, and Release Time
- Interrupts, interrupts mask, interrupts clear, interrupts status
- Multi-key detect
- FIFO with width of 12 bits and depth of 16 to store Key Press and Release Events
- Two work modes: Event (Press and Release) Trigger Mode and Regular Scan Mode (high priority)
- Low power mode: Key press event can wakeup CPU from sleep

16.2 Functional Description

16.2.1 Block Diagram

The block diagram of Key-Scan is shown in Fig 16-1.

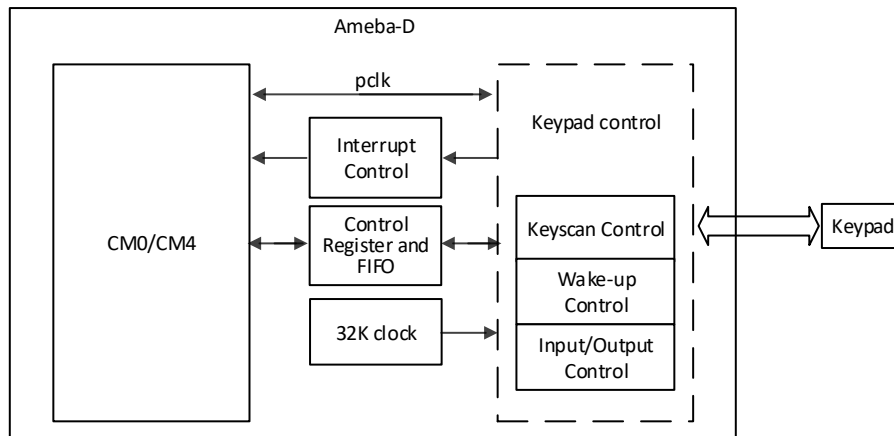


Fig 16-1 Key-Scan block diagram

The typical application setup is shown in Fig 16-2 (take 6*6 keypad array for example), external keypad is needed.

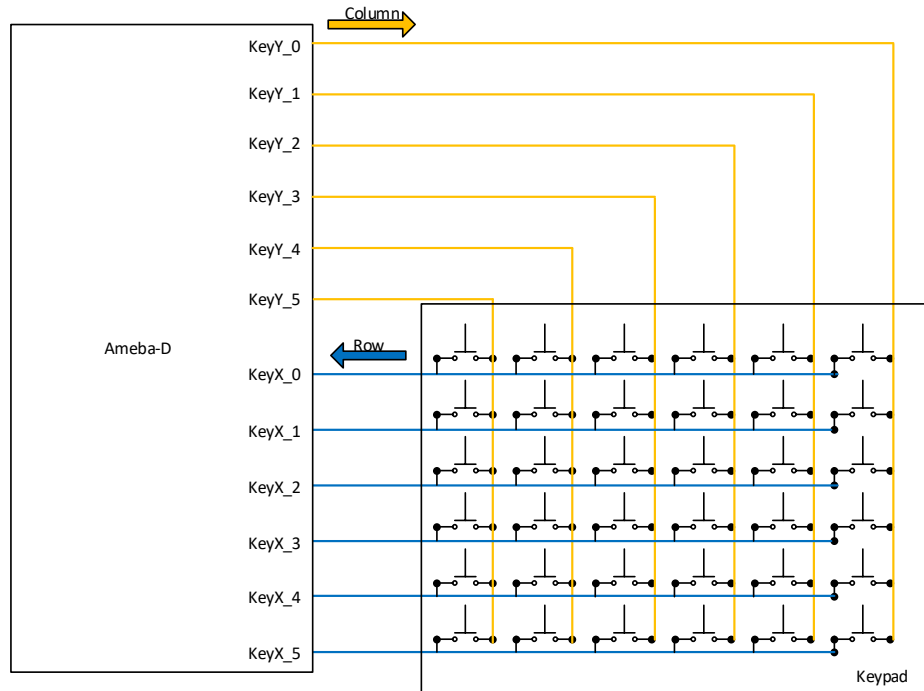


Fig 16-2 Typical application setup with external keypad

16.2.2 Work Principle

At the beginning, all columns output low, and all rows are set as input with external pull up.

When there is a key or multiple keys pressed (Short between Column and Row), it triggers an internal state machine to start a Key-Scan cycle to determine the column and row of the key that is pressed. The state machine sets first column as an output low and all other columns are in Hi-Z state. The state machine then scans all rows to determine which keys are being pressed, and then second column is output low until the last column. Firstly, scan circuit waits for a first debounce time and does first scan. Then, the circuit waits for a debounce time and scans again. After the two scan cycles, the circuit decides the current key status, and compares with internal key status table to find which key is pressed or released. After a full scan, the key press or release events are stored in the key event FIFO. Debounce time for the keypad is for each full scan and is configurable, as shown in Fig 16-3.

Interval time between each full scan is configurable. Once all keys are detected release, there is a release timer to confirm that, then an all release interrupt is triggered and state machine enters idle state. The Key-Scan timing is shown in Fig 16-4.

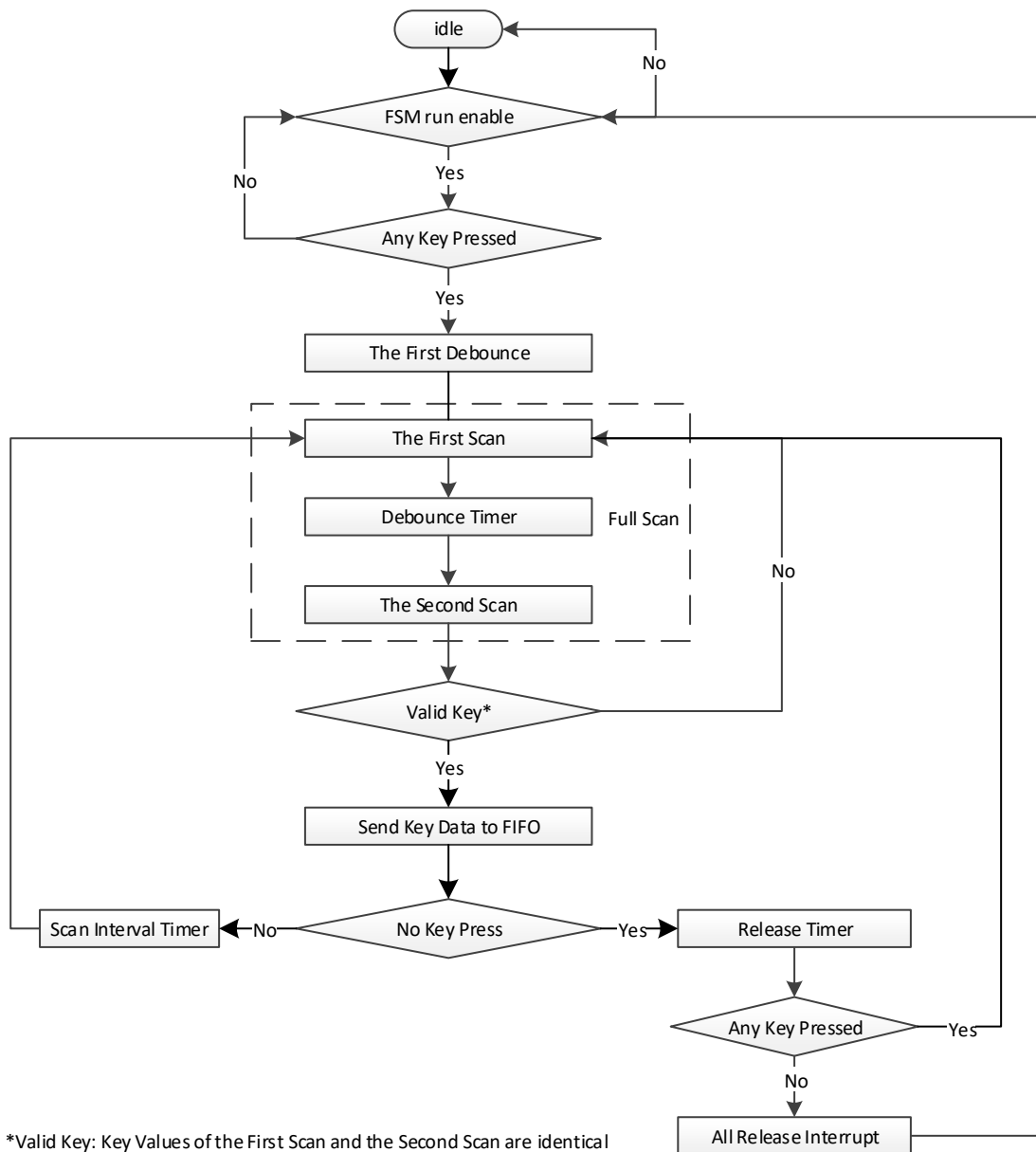


Fig 16-3 Key-Scan flow

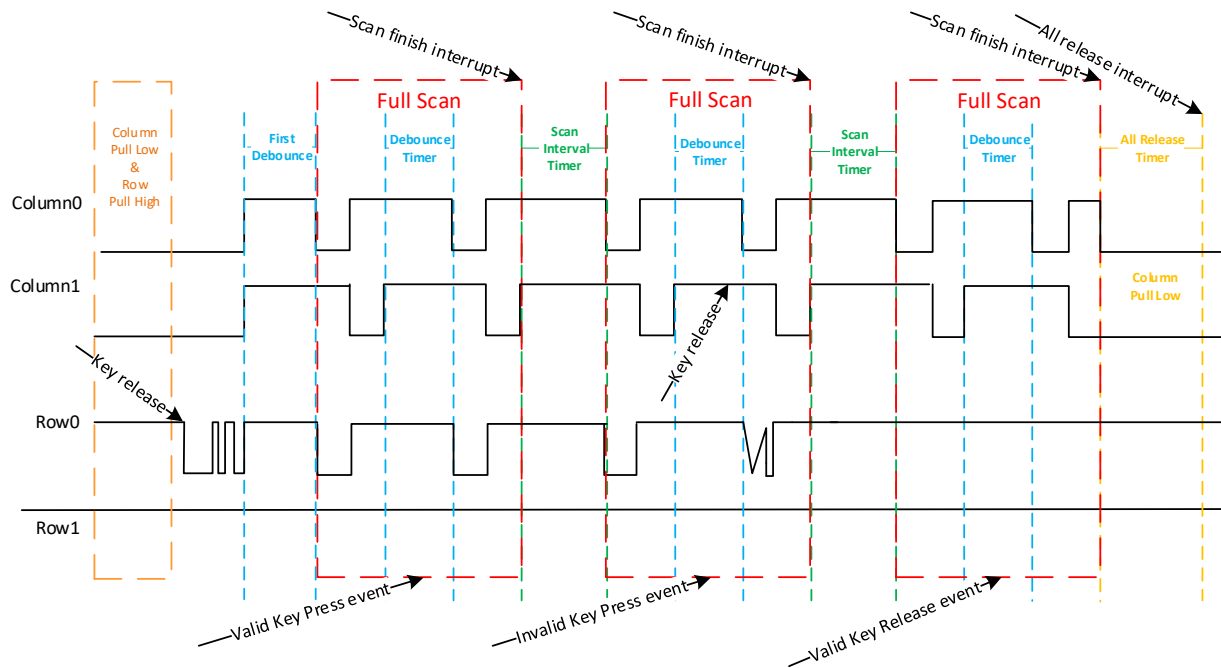


Fig 16-4 Key-Scan timing

There are two work modes provided: Event Trigger Mode and Regular Scan Mode.

16.2.2.1 Event Trigger Mode

In Event Trigger Mode, key press or release event is stored in the key event FIFO only once in each key press and release operation.

16.2.2.2 Regular Scan Mode

In Regular Scan Mode, at each full scan, any key press event is stored in the key event FIFO until it is released (in this condition, only key press event is stored in the key event FIFO). Fig 16-5 shows the difference of FIFO items between Event Trigger Mode and Regular Scan Mode during a key press and release.

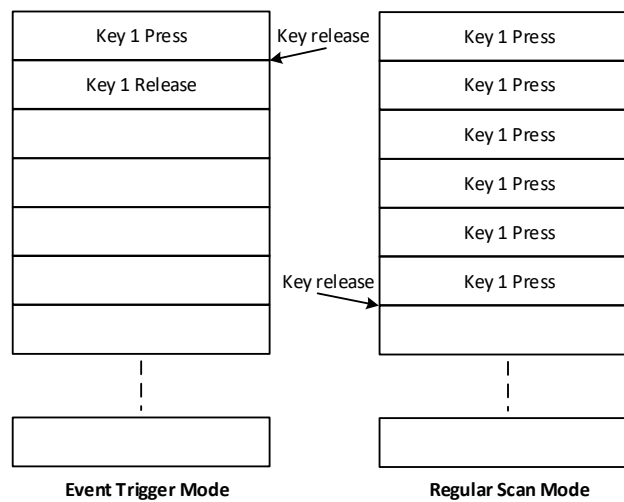


Fig 16-5 Difference of FIFO items between two work modes

16.2.3 FIFO Mechanism

The FIFO with depth of 16 and width of 12-bit is used to store key press and release events, as shown in Fig 16-6.

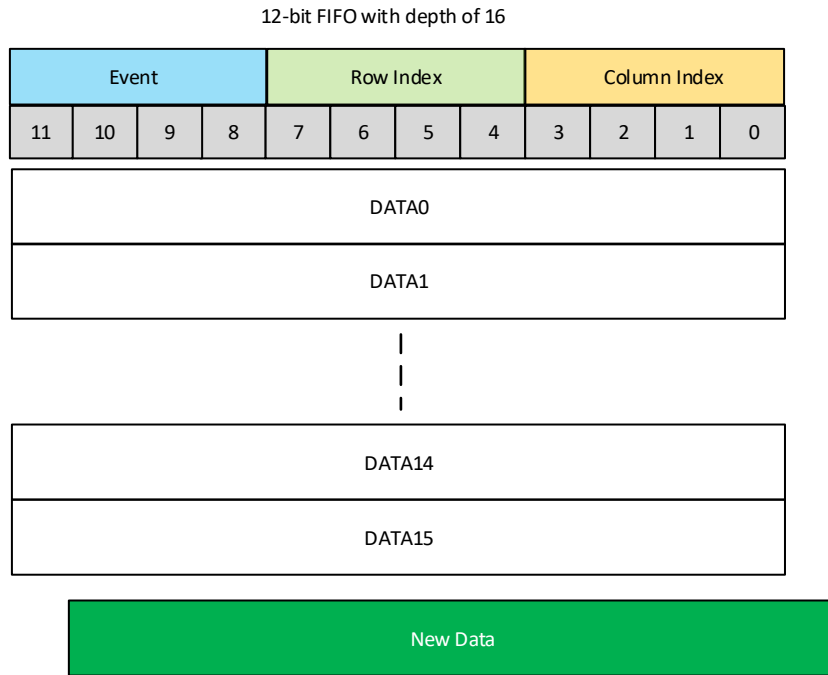


Fig 16-6 FIFO structure

In the FIFO item, bit[7:0] indicates the key # pressed or released in a keypad array. Bit[8] indicates a key press or release event happened. A '0' means a key release event. A '1' means that a key has been pressed (which can be cleared on a read). When FIFO is full and another item is pushed into the FIFO, an overflow interrupt is triggered. When FIFO is empty and is read, an overread interrupt is triggered.

The key value assignment is listed in Table 16-1.

Table 16-1 Key value assignment

Column \ Row	0	1	2	3	4	5	6	7
0	11	12	13	14	15	16	17	18
1	21	22	23	24	25	26	27	28
2	31	32	33	34	35	36	37	38
3	41	42	43	44	45	46	47	48
4	51	52	53	54	55	56	57	58
5	61	62	63	64	65	66	67	68
6	71	72	73	74	75	76	77	78
7	81	82	83	84	85	86	87	88

16.2.4 Clock Configuration

There are two clock domains in the Key-Scan design, as shown in Fig 16-7.

- The green part works in low power clock domain, which clock isn't gated in low power modes.
- The blue part works under bus clock domain (10M), it is gated in low power modes. In bus clock domain, some of the logic works in scan_clk frequency, which is divided from bus clock.

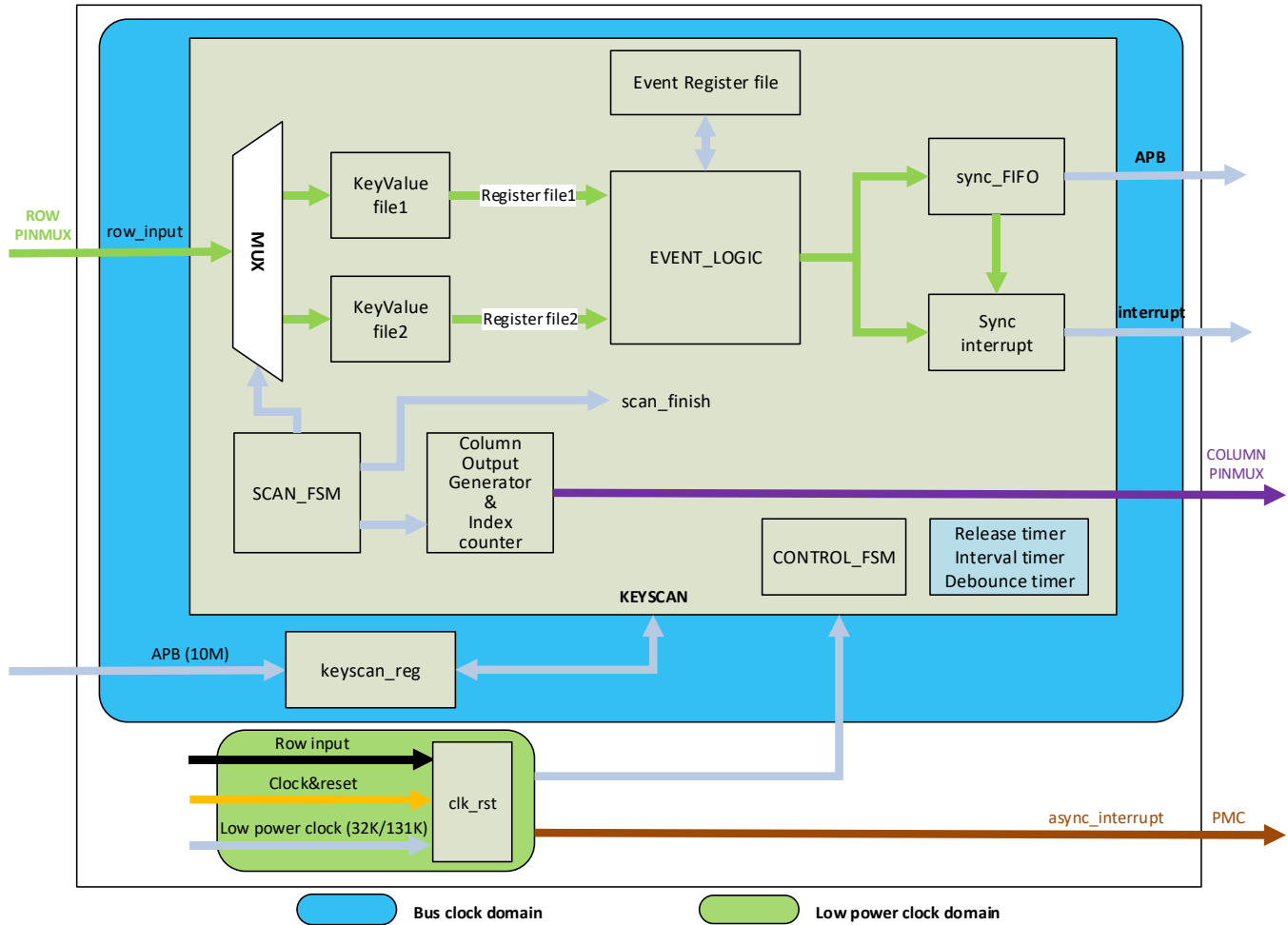


Fig 16-7 Clock domain diagram

The Key-Scan trigger signal is generated by lower frequency (32K/131K). When low power clock detected a key press, hardware starts scan by scan clock to confirm the key values.

16.2.5 Shadow Key Problem

Ameba-D supports multiple key presses detect accurately. Applications requiring three-key combinations (such as <Ctrl><Alt>, or any other combinations) must ensure that the three keys are wired in appropriate key positions to avoid shadow key (or appearing like a 4th key has been pressed).

To avoid shadow key, it is best to keep 3-button combinations that are pressed on separate rows and columns. Consider the situation with the keypad described in Fig 16-8.

In the keypad setup in Fig 16-8, there is a 4*3 keypad matrix, connected to Row0 ~ Row3, and Column0 ~ Column2. All of the rows are configured as inputs with pull up resistors. The columns are configured as outputs, driving low. When a key press is made, one of the row inputs is pulled low, letting the RTL8721D know that a key has been pressed, and the RTL8721D then starts the key scanning algorithm. During this algorithm, it sweeps the output low across the columns, such that only 1 column is driven low at a time. While this is done to each column, the RTL8721D reads the row inputs, to determine which keys on a column are being pressed.

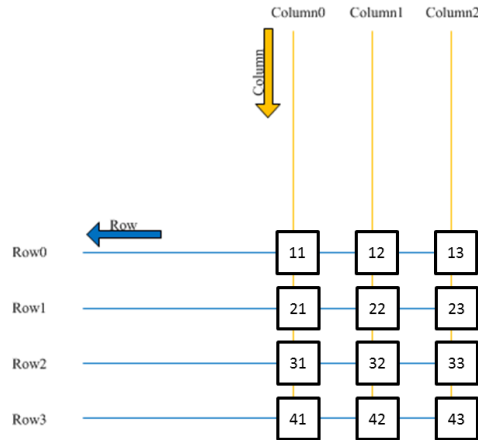


Fig 16-8 4*3 keypad example

Shadow key can occur when multiple keys are pressed that can make it appear that additional keys (which are not being pressed) are being pressed. In Fig 16-9, keys 11, 12, and 21 are pressed, which causes a shadow key issue. Since Row1 becomes pulled to ground through key 11 (which is pulled through key 12 when Column1 is transmitting a low), when Column1 is driving low, the RTL8721D sees a low signal at both Row0 and Row1. This falsely triggers key 22 as being pressed (the key highlighted as yellow).

The reason for this is that keypad matrices short the columns to the rows connected together. When Column1 is driving low, the low gets transmitted onto Row0 via key 12. Key 11 is being pressed, which also shorts Column0 to ground. Key 11 is pressed, which then shorts Row1 to Column0. In this process, Row1 is shorted to Column1, which is the reason shadow Key occurs.

Keypad matrices can support multiple key presses properly, if care is taken when choosing the layout. In Fig 16-10, you see a 3-button combination which works as expected.

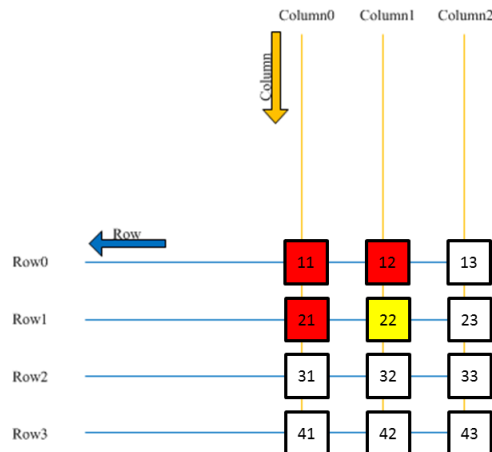


Fig 16-9 Shadow key condition

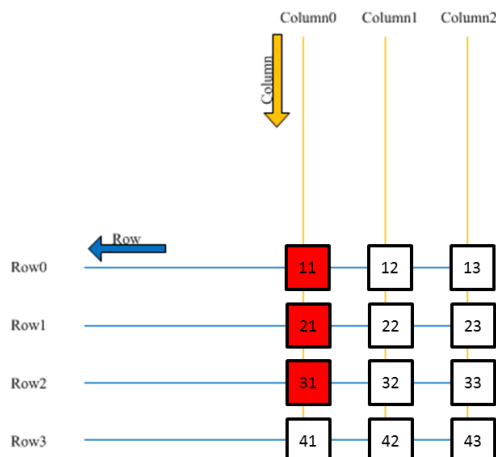


Fig 16-10 Correct three-key condition

16.3 Registers

The details of Key-Scan register are listed in Table 16-2. The base address of Key-Scan is 0x4800_A000.

Table 16-2 Register map of Key-Scan

Name	Offset	Access	Description
KS_CLK_DIV	0x00	R/W	Configure Scan clock
KS_TIM_CFG0	0x04	R/W	Configure corresponding Timer (pre/post guard timer, debounce timer)
KS_TIM_CFG1	0x08	R/W	Configure corresponding Timer (interval timer, all release timer)
KS_CTRL	0x0C	R/W	Enable Key-Scan and interrupt
KS_FIFO_CFG	0x10	R/W	FIFO limit level and threshold level setup
KS_COL_CFG	0x14	R/W	Configure keypad columns
KS_ROW_CFG	0x18	R/W	Configure keypad rows
KS_DATA_NUM	0x1C	R	Number of data in FIFO
KS_DATA	0x20	R	Indicate key # and key event
KS_IMR	0x24	R/W	Configure masking of Key-Scan interrupt
KS_ICR	0x28	W1C	Write 1 to clear Key-Scan active interrupt
KS_ISR	0x2C	R	Return the masked Key-Scan interrupts after masking with the KS_IMR register
KS_ISR_RAW	0x30	R	Return the raw Key-Scan interrupts
KS_DUMMY	0x34	R/W	Configure interval polarity and enable the discharge of column

16.3.1 KS_CLK_DIV

- **Name:** Key-Scan Clock Division Register
- **Size:** 32 bits
- **Address offset:** 0x00
- **Read/write access:** read/write

31	30	29	...	14	13	12	11	10	9	...	2	1	0
RSVD							KS_CLK_DIV						
							R/W						

Bit	Name	Access	Reset	Description
31:12	RSVD	N/A	-	Reserved
11:0	KS_CLK_DIV	R/W	0x0	scan_clk = bus clock/(KS_CLK_DIV +1)

If bus clock = 10M, scan_clk = 2.44kHz ~ 10MHz, 999 clock division is suggested to generate 10k scan clock.

16.3.2 KS_TIM_CFG0

- **Name:** Key-Scan Timer Configuration 0 Register
- **Size:** 32 bits
- **Address offset:** 0x04
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				KS_POST_GUARD_TIMER				RSVD				KS_PRE_GUARD_TIMER			
				R/W								R/W			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				KS_DEB_TIMER											
				R/W											

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	--	Reserved
27:24	KS_POST_GUARD_TIMER	R/W	0x3	Post guard time for key column post guard time = scan_clk * KS_POST_GUARD_TIMER
23:20	RSVD	N/A	-	Reserved
19:16	KS_PRE_GUARD_TIMER	R/W	0x3	Pre guard time for key column pre guard time = scan_clk * KS_PRE_GUARD_TIMER
15:12	RSVD	N/A	-	Reserved
11:0	KS_DEB_TIMER	R/W	0x0	Debounce timer debounce timer = scan_clk * (KS_DEB_TIMER + 1) Debounce timer ranges from 100ns to 1.68s, and 5ms is suggested.

16.3.3 KS_TIM_CFG1

- **Name:** Key-Scan Timer Configuration 1 Register
- **Size:** 32 bits
- **Address offset:** 0x08
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				KS_INTERVAL_TIMER											
				R/W											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				KS_RELEASE_TIMER											
				R/W											

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27:16	KS_INTERVAL_TIMER	R/W	0x0	Scan interval timer interval timer = scan_clk * (KS_INTERVAL_TIMER + 1) Interval timer ranges from 100ns to 1.68s.
15:12	RSVD	N/A	-	Reserved
11:0	KS_RELEASE_TIMER	R/W	0x0	Release detect timer release timer = scan_clk * (KS_RELEASE_TIMER + 1) Release timer ranges from 100ns to 1.68s.

16.3.4 KS_CTRL

- **Name:** Key-Scan Control Register

- **Size:** 32 bits
- **Address offset:** 0x0C
- **Read/write access:** read/write

31	30	...	5	4	3	2	1	0
RSVD					KS_WORK_MODE	RSVD	KS_BUSY_STATUS	KS_RUN_ENABLE
					R/W		RO	R/W

Bit	Name	Access	Reset	Description
31:4	RSVD	N/A	-	Reserved
3	KS_WORK_MODE	R/W	0x0	Work mode <ul style="list-style-type: none"> ● 0x1: Event Trigger Mode ● 0x0: Regular Scan Mode
2	RSVD	N/A	-	Reserved
1	KS_BUSY_STATUS	RO	0x0	FSM busy status
0	KS_RUN_ENABLE	R/W	0x0	Enable Key-Scan internal scan clock The Key-Scan clock must be enabled after Key-Scan configuration is done. <ul style="list-style-type: none"> ● 0x1: Enable Key-Scan clock ● 0x0: Disable Key-Scan clock

16.3.5 KS_FIFO_CFG

- **Name:** Key-Scan FIFO Configuration Register
- **Size:** 32 bits
- **Address offset:** 0x10
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				KS_FIFO_LIMIT_LEVEL				RSVD				KS_FIFO_TH_LEVEL			
				R/W								R/W			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													KS_FIFO_OV_CTRL	KS_FIFO_CLR	
													R/W	W1C	

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	-	Reserved
27:24	KS_FIFO_LIMIT_LEVEL	R/W	0x0	Limit the max allowable key number be pressed at a time <ul style="list-style-type: none"> ● 0x0: No limit ● 0x1: Only one pressed event is allowable at a time ● ... ● 0x6: Max 6 pressed event is allowable at a time ● 0x7 ~ 0xf: DO NOT USE
23:20	RSVD	N/A	-	Reserved
19:16	KS_FIFO_TH_LEVEL	R/W	0x0	FIFO threshold setup
15:2	RSVD	N/A	-	Reserved
1	KS_FIFO_OV_CTRL	R/W	0x0	FIFO overflow control <ul style="list-style-type: none"> ● 0x0: Rejects the new scan data when FIFO is totally full ● 0x1: Discards the oldest scan data when FIFO is totally full
0	KS_FIFO_CLR	W1C	-	Write 1 to clear FIFO data

16.3.6 KS_COL_CFG

- **Name:** Key-Scan Column Configuration Register
- **Size:** 32 bits
- **Address offset:** 0x14

- **Read/write access:** read/write

31	30	29	28	27	26	...	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													KS_COL_SEL							
													R/W							

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7:0	KS_COL_SEL	R/W	0x0	Define which column is used <ul style="list-style-type: none"> ● 0b00000000: No key column is selected ● 0b00000001: Key column 0 is selected ● 0b00000010: Key column 1 is selected ● 0b00000011: Key column 0 and column 1 are selected ● 0b00000100: Key column 2 is selected ● ...

16.3.7 KS_ROW_CFG

- **Name:** Key-Scan Row Configuration Register
- **Size:** 32 bits
- **Address offset:** 0x18
- **Read/write access:** read/write

31	30	29	28	27	26	...	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													KS_ROW_SEL							
													R/W							

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7:0	KS_ROW_SEL	R/W	0x0	Define which row is used <ul style="list-style-type: none"> ● 0b00000000: No key row is selected ● 0b00000001: Key row 0 is selected ● 0b00000010: Key row 1 is selected ● 0b00000011: Key row 0 and row 1 are selected ● 0b00000100: Key row 2 is selected ● ...

16.3.8 KS_DATA_NUM

- **Name:** Key-Scan FIFO Data Number Register
- **Size:** 32 bits
- **Address offset:** 0x1C
- **Read/write access:** read-only

31	30	29	28	...	21	20	19	18	17	16
RSVD									KS_FIFO_FULL	KS_FIFO_EMPTY
									R	R
15	14	13	...	6	5	4	3	2	1	0
RSVD						KS_FIFO_DATA_LEVEL				
						R				

Bit	Name	Access	Reset	Description
31:18	RSVD	N/A	-	Reserved
17	KS_FIFO_FULL	R	0x0	FIFO full status
16	KS_FIFO_EMPTY	R	0x1	FIFO empty status
15:5	RSVD	N/A	-	Reserved

4:0	KS_FIFO_DATA_LEVEL	R	0x0	Number of entry in FIFO
-----	--------------------	---	-----	-------------------------

16.3.9 KS_DATA

- **Name:** Key-Scan Event FIFO Data Register
- **Size:** 32 bits
- **Address offset:** 0x20
- **Read/write access:** read-only
-

When reading this register, these bits return the value in the Event FIFO.

31	30	29	28	...	15	14	13	12	11	10	9	...	2	1	0
RSVD									KS_DATA						
									R						

Bit	Name	Access	Reset	Description
31:12	RSVD	N/A	-	Reserved
11:0	KS_DATA	R	0x0	<ul style="list-style-type: none"> ● Bit[11:8]: Key press or release event <ul style="list-style-type: none"> ■ 0x0: Key release event ■ 0x1: Key press event ● Bit[7:4]: Row index ● Bit[3:0]: Column index

16.3.10 KS_IMR

- **Name:** Key-Scan Interrupt Mask Register
- **Size:** 32 bits
- **Address offset:** 0x24
- **Read/write access:** read/write

These bits mask their corresponding interrupt status bits. This register is active low; a value of 0 masks the interrupt, whereas a value of 1 unmask the interrupt.

31	30	29	...	9	8	7
RSVD						
6	5	4	3	2	1	0
KS_SCAN_EVENT_INT_MASK	KS_FIFO_LIMIT_INT_MASK	KS_FIFO_OV_INT_MASK	KS_FIFO_FULL_INT_MASK	KS_SCAN_FINISH_INT_MASK	KS_FIFO_NOTEMPTY_INT_MASK	KS_ALL_RELEASE_INT_MASK
R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:7	RSVD	N/A	-	Reserved
6	KS_SCAN_EVENT_INT_MASK	R/W	0x0	<ul style="list-style-type: none"> ● 0x0: Mask scan event interrupt ● 0x1: Unmask scan event interrupt
5	KS_FIFO_LIMIT_INT_MASK	R/W	0x0	<ul style="list-style-type: none"> ● 0x0: Mask FIFO limit interrupt ● 0x1: Unmask FIFO limit interrupt
4	KS_FIFO_OV_INT_MASK	R/W	0x0	<ul style="list-style-type: none"> ● 0x0: Mask FIFO overflow interrupt ● 0x1: Unmask FIFO overflow interrupt
3	KS_FIFO_FULL_INT_MASK	R/W	0x0	<ul style="list-style-type: none"> ● 0x0: Mask FIFO full interrupt ● 0x1: Unmask FIFO full interrupt
2	KS_SCAN_FINISH_INT_MASK	R/W	0x0	<ul style="list-style-type: none"> ● 0x0: Mask scan finish interrupt ● 0x1: Unmask scan finish interrupt
1	KS_FIFO_NOTEMPTY_INT_MASK	R/W	0x0	<ul style="list-style-type: none"> ● 0x0: Mask FIFO non-empty interrupt ● 0x1: Unmask FIFO non-empty interrupt
0	KS_ALL_RELEASE_INT_MASK	R/W	0x0	<ul style="list-style-type: none"> ● 0x0: Mask all release interrupt

				● 0x1: Unmask all release interrupt
--	--	--	--	-------------------------------------

16.3.11 KS_ICR

- **Name:** Key-Scan Interrupt Clear Register
- **Size:** 32 bits
- **Address offset:** 0x28
- **Read/write access:** write

The entire interrupt bits read as zero and write '1' to clear the corresponding interrupt.

31	30	...	8	7	6	5	4	3	2	1	0
RSVD						KS_FIFO_LIMIT_INT_CLR	KS_FIFO_OV_INT_CLR	RSVD	KS_SCAN_FINISH_INT_CLR	RSVD	KS_ALL_RELEASE_INT_CLR
						W1C	W1C		W1C		W1C

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	-	Reserved
5	KS_FIFO_LIMIT_INT_CLR	W1C	-	Clear FIFO limit interrupt flag Write '1' to clear.
4	KS_FIFO_OV_INT_CLR	W1C	-	Clear FIFO overflow interrupt flag Write '1' to clear.
3	RSVD	N/A	-	Reserve bit, because this interrupt is automatically cleared by hardware.
2	KS_SCAN_FINISH_INT_CLR	W1C	-	Clear scan finish interrupt flag Write '1' to clear.
1	RSVD	N/A	-	Reserve bit, because this interrupt is automatically cleared by hardware.
0	KS_ALL_RELEASE_INT_CLR	W1C	-	Clear all release interrupt flag Write '1' to clear.

16.3.12 KS_ISR

- **Name:** Key-Scan Masked Interrupt Status Register
- **Size:** 32 bits
- **Address offset:** 0x2C
- **Read/write access:** read-only

31	30	29	...	9	8	7
RSVD						
6	5	4	3	2	1	0
KS_SCAN_EVENT_INT_STATUS	KS_FIFO_LIMIT_INT_STATUS	KS_FIFO_OV_INT_STATUS	KS_FIFO_FULL_INT_STATUS	KS_SCAN_FINISH_INT_STATUS	KS_FIFO_NOTEMPTY_INT_STATUS	KS_ALL_RELEASE_INT_STATUS
R	R	R	R	R	R	R

Bit	Name	Access	Reset	Description
31:7	RSVD	N/A	-	Reserved
6	KS_SCAN_EVENT_INT_STATUS	R	0x0	Masked scan event interrupt status
5	KS_FIFO_LIMIT_INT_STATUS	R	0x0	Masked FIFO limit interrupt status
4	KS_FIFO_OV_INT_STATUS	R	0x0	Masked FIFO overflow interrupt status
3	KS_FIFO_FULL_INT_STATUS	R	0x0	Masked FIFO full interrupt status
2	KS_SCAN_FINISH_INT_STATUS	R	0x0	Masked scan finish interrupt status
1	KS_FIFO_NOTEMPTY_INT_STATUS	R	0x0	Masked FIFO nonempty interrupt status
0	KS_ALL_RELEASE_INT_STATUS	R	0x0	Masked all release interrupt status

16.3.13 KS_ISR_RAW

- **Name:** Key-Scan Raw Interrupt Status Register
- **Size:** 32 bits
- **Address offset:** 0x30
- **Read/write access:** read-only

31	30	29	...	9	8	7
RSVD						
6	5	4	3	2	1	0
KS_SCAN_EVENT_RAW_INT_STATUS	KS_FIFO_LIMIT_RAW_INT_STATUS	KS_FIFO_OV_RAW_INT_STATUS	KS_FIFO_FULL_RAW_INT_STATUS	KS_SCAN_FINISH_RAW_INT_STATUS	KS_FIFO_NOTEMPTY_RAW_INT_STATUS	KS_ALL_RELEASE_RAW_INT_STATUS
R	R	R	R	R	R	R

Bit	Name	Access	Reset	Description
31:7	RSVD	N/A	-	Reserved
6	KS_SCAN_EVENT_RAW_INT_STATUS	R	0x0	Scan event raw interrupt status KS_SCAN_EVENT_RAW_INT_STATUS is triggered when there is one or more scan event(s) sent into FIFO in a full scan. It is automatically cleared when the event is read by the KS_DATA register.
5	KS_FIFO_LIMIT_RAW_INT_STATUS	R	0x0	FIFO limit raw interrupt status KS_FIFO_LIMIT_RAW_INT_STATUS is triggered when there is more than KS_FIFO_LIMIT_LEVEL keys are pressed at a time. For Event Trigger Mode, the pressed key isn't sent into FIFO; and for Regular Scan Mode, the first scanned keys with number KS_FIFO_LIMIT_LEVEL are sent into FIFO.
4	KS_FIFO_OV_RAW_INT_STATUS	R	0x0	FIFO overflow raw interrupt status KS_FIFO_OV_RAW_INT_STATUS is triggered when FIFO is completely full (16 entries), and there is another event is sent into it. At this time, whether to reject the new scan data or discard the oldest scan data is controlled by KS_FIFO_OV_CTRL.
3	KS_FIFO_FULL_RAW_INT_STATUS	R	0x0	FIFO full raw interrupt status KS_FIFO_FULL_RAW_INT_STATUS is triggered when the number of events in the FIFO reaches or goes above the threshold in the KS_FIFO_TH_LEVEL. It is automatically cleared when the data level goes below the threshold.
2	KS_SCAN_FINISH_RAW_INT_STATUS	R	0x0	Scan finish raw interrupt status KS_SCAN_FINISH_RAW_INT_STATUS is triggered when each full scan finished whether there is any key press or release events occur or not.
1	KS_FIFO_NOTEMPTY_RAW_INT_STATUS	R	0x0	FIFO non-empty raw interrupt status KS_FIFO_NOTEMPTY_RAW_INT_STATUS is triggered when there are events in the FIFO. It is automatically cleared when the FIFO is empty.
0	KS_ALL_RELEASE_RAW_INT_STATUS	R	0x0	All release raw interrupt status KS_ALL_RELEASE_RAW_INT_STATUS is triggered when the release time counter reaches the value set in the KS_RELEASE_TIMER and there is no more keys pressed.

16.3.14 KS_DUMMY

- **Name:** Key-Scan Interval Polarity and Discharge Configuration Register
- **Size:** 32 bits
- **Address offset:** 0x34
- **Read/write access:** read/write

31	30	29	...	18	17	16
----	----	----	-----	----	----	----

RSVD															
15		14		13		12		11		10		9		8	
DUMMY_H															
R/W															
7		6		5		4		3		2		1		0	
DUMMY_L												KS_DISCHARGE		KS_INTERVAL_P OLARITY	
R/W												R/W		R/W	

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:8	DUMMY_H	R/W	0xFF	Dummy_h
7:2	DUMMY_L	R/W	0x0	Dummy_l
1	KS_DISCHARGE	R/W	0x0	Discharge the column spurious capacitance <ul style="list-style-type: none"> 0x1: Enable the discharge of the column spurious capacitance for two scan cycle. 0x0: Disable the discharge of the column spurious capacitance.
0	KS_INTERVAL_POLARITY	R/W	0x0	Configure the column polarity in debounce and interval phase <ul style="list-style-type: none"> 0x1: Drive low 0x0: Open drain floating

17 Audio Codec (AC)

17.1 Introduction

Ameba-D audio codec is a stereo audio codec with stereo headphone amplifiers, as well as 2-way inputs and 1-way stereo/mono output that are programmable to single end or differential.

Ameba-D audio codec integrates anti-pop circuit for audible pop noise cancellation, mic bias circuit for mic power supply and programmable mic boost ability.

The digital part of audio codec IP implements silence detection, side-tone playback, volume select, digital volume control, 5-band equalizer, etc.

Ameba-D audio codec supports the highest 96kHz sampling frequency and 24-bit resolution, while transmits record data to or receives playback data from platform through I²S bus. Ameba-D platform configures and drives audio codec by specific serial interface (SI).

Besides the internal audio codec IP, Ameba-D also provides external I²S interfaces for audio application extension, which support the highest 384kHz sampling frequency.

17.2 Diagram

The diagram of audio codec is shown in Fig 17-1.

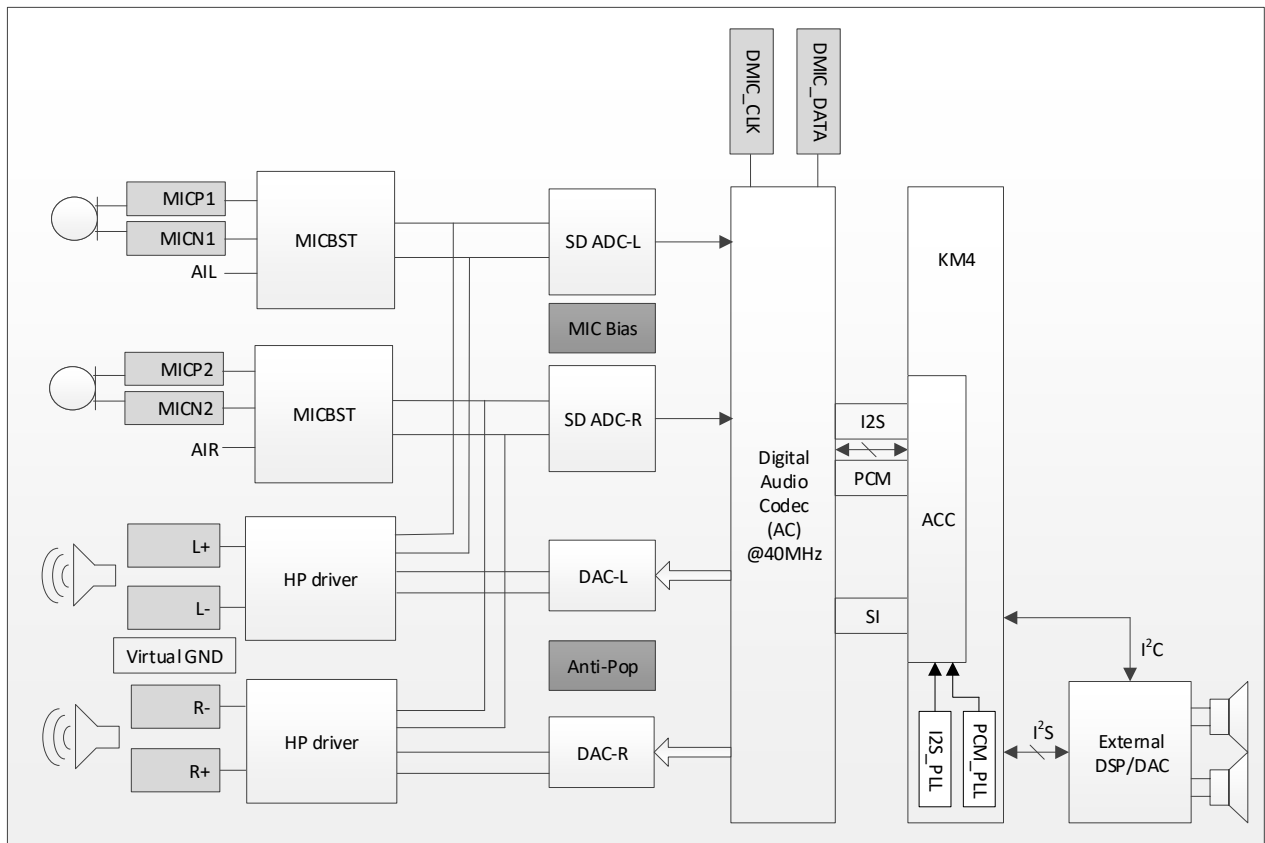


Fig 17-1 Audio codec diagram

17.3 Key Features

17.3.1 Analog Part

The analog key features are listed in Table 17-1.

Table 17-1 Analog key features

Items	Features
Internal LDO	Internal LDO output AVCC 2.8V or 1.7V
Separate power down modes for ADC and DAC	Shutdown mode Standby (All clocks are OFF except R/W registers) <ul style="list-style-type: none"> ● L/R output driver power-down individually ● L/R ADC power-down individually ● L/R DAC power-down individually
Anti-pop function to reduce audible pop	Pop reduction during power on/off/mute Cap-less/single-ended/differential modes No audible pop is required
Built-in circuit	MIC bias Reference and biasing circuitry
Line-in options	Line-in signal to ADC for silence detection, EQ, and gain control Line-in signal to headphone driver without gain control
Input/Output	2 mono differential or single-end mic inputs or 1 stereo single-ended line input 1 stereo single-ended output or 1 stereo cap-less output or 1 stereo differential output
ADC	SNR: 88dB for line input to ADC path @AVCC=2.8V THD+N: -80dB for line input to ADC path @-3dBFS Dynamic range: 92dB 2nd order and 3 bits
DAC	SNR: 95dB for DAC to earphone output @AVCC=2.8V THD+N: <ul style="list-style-type: none"> ● 0.01% @40mW/16Ω for DAC to earphone output ● 0.005% @FS-1dB/10KΩ for DAC to line output Dynamic range: 96dB 2nd order and 4 bits
PSRR (Power Supply Rejection Ratio)	<ul style="list-style-type: none"> ● 60dB @20~200Hz ● 70dB @200~20kHz
ADC input gain range	Gain control by software
MIC input boost gain stage	0/20/30/40dB
Earphone output gain range	Gain control by software
Earphone typical output power @3.3V	<ul style="list-style-type: none"> ● 40mW on 16Ω load ● 20mW on 32Ω load
Sampling frequency	8/16/32/44.1/48/88.2/96kHz
Main clock	40MHz

17.3.2 Digital Part

The digital key features are listed in Table 17-2.

Table 17-2 Digital key features

Items	Features
I ² S/PCM Interface	Slave mode ASRC is required for ADC/DAC paths 24 bits information for ADC and DAC paths
Sampling Frequency	8/16/32/44.1/48/88.2/96kHz
Over-Sampling Rate	<ul style="list-style-type: none"> ● OSR = 128 * fs

	<ul style="list-style-type: none"> ● CLK rate > 256 * fs
Main Clock Rate	MCLK = 40MHz <ul style="list-style-type: none"> ● 40MHz for 88.2/96kHz ● 20MHz for 44.1/48kHz ● 10MHz for 8/16/32kHz
SDM Clock Rate	<ul style="list-style-type: none"> ● 20MHz for 88.2/96kHz ● 10MHz for 44.1/48kHz ● 5MHz for 8/16/32kHz
Anti-pop function to reduce audible pop	Pop reduction during power on/off/mute
FIR filters in DAC path	Linear-phase filter
MUX options in DAC and ADC paths	L/R mixing (L+R→L/R), mute (0→L/R) L/R swap (L→L/R, R→L/R) Karaoke mode mixing
ADC Path	Side-tone generator DC remover (adjustable High Pass Filter) 5-band bi-quad audio EQ (24-bit EQ coefficients) for MIC frequency response compensation ASRC (Asynchronous Sampling Rate Converter) is required if acting as a I2S/PCM slave device Silence detection @ADC path
Gain Control in ADC Path	Gain Step: 0.375dB/step Gain Range: -17.625 dB ~30dB 7-bit programmable gain
DAC Path	ASRC (Asynchronous Sampling Rate Converter) is required if acting as a I2S/PCM slave device 5-band bi-quad EQ for speaker frequency response compensation DC remover (HPF), Dither One full-band DRC (ALC)
Gain Control in DAC Path	Gain Step: 0.375dB/step Gain Range: -64.5dB ~ 0dB 8-bit programmable gain
ASRC (Asynchronous Sampling Rate Converter)	Auto tracking mode (default) Programmable tracking mode
Digital MIC Interface	One DMIC interface Boost

17.4 Specifications

17.4.1 DAC Path

The DAC path electrical characteristics are illustrated in Table 17-3.

Table 17-3 DAC path electrical characteristics

Item	Conditions	Minimum	Typical	Maximum	Unit
Over-sampling rate (f)			128		f _s
Resolution			16		Bits
Output Sample Rate, F _{sample}	8,16,32,44.1,48,88.2,96kHz	8		96	kHz
Signal to Noise Ratio (SNR @cap-less mode)	f _{in} =1kHz B/W=20~20kHz THD+N < 0.01% 0dBFS signal Load=10kΩ	AVCC=2.8V	95		dB
Signal to Noise Ratio (SNR @single-end mode)	f _{in} =1kHz B/W=20~20kHz THD+N < 0.01% 0dBFS signal Load=10kΩ	AVCC=2.8V	90		dB
Digital Gain		-65.625		0	dB

Digital Gain Resolution			0.375		dB
Output Voltage Full-scale Swing	$X = 0.8 \cdot AVCC / 2.828 (V_{RMS})$	AVCC=1.70V	509 (1.44)		mV _{RMS}
		AVCC=2.8V	800 (2.24)		(V _{PP})
Maximum Output Power	16Ω load	AVCC=1.70V	16		mW
		AVCC=2.8V	40		
	32Ω load	AVCC=1.70V	8		
		AVCC=2.8V	20		
Allowed Load	Resistive		16	O.C.	Ω
	Capacitive	1500			pF
THD+N	100KΩ load	0.005		0.01	%
	16Ω load	0.01		0.05	
Signal to Noise Ratio (SNR @ earphone load)	16Ω load, 0dBFS input relative to digital silence		96		dB
L/R output variation	L vs. R, measured at -10dBFS@1kHz input		±0.1 (< 1%)	±0.2	dB
L/R output variation (IC by IC)			±0.2 (< 2%)		dB
Crosstalk between channels	L vs. R, measured at -10dBFS@1kHz input		-80		dB
Power ON pop noise	Cap-less mode			-66	dBV
	Single-ended mode			-60 (1mV _{RMS})	dBV
	Differential mode			-66	dBV
Analog supply voltage (AVCC)	<ul style="list-style-type: none"> For earphone: AVCC=1.7V For earphone/speakers: AVCC=2.8V 	1.7	2.8	3.0	V
Digital supply voltage (DVDD)		0.99	1.1	1.21	V

17.4.2 ADC Path

The ADC path electrical characteristics are illustrated in Table 17-4.

Table 17-4 ADC path electrical characteristics

Item	Conditions	Minimum	Typical	Maximum	Unit
Resolution			16		Bits
Input sample rate, F _{sample}	8/16/32/44.1/48/ 88.2/96kHz Note: Better to have 88.2/96kHz	8		96	kHz
Signal to Noise Ratio (SNR @Line-in mode)	f _{in} =1kHz B/W=20~20kHz THD+N < 0.1% 800mV _{RMS} input (AVCC=2.8V)	8kHz	88		dB
		16kHz	88		
		32kHz	88		
		44.1kHz	88		
		48kHz	88		
Signal to Noise Ratio (SNR @MIC mode)	f _{in} =1kHz B/W=20~20kHz THD+N < 0.5% 800mV _{RMS} input 40dB gain (AVCC=2.8V)	8/16kHz	87		dB
Digital Gain		-17.625		30	dB
Digital Gain Resolution			0.375		dB
MIC Boost Gain	0/20/30/40dB	0	20	40	dB
Input full-scale at maximum gain (differential)	Gain=40dB		2.828*AVCC		mV _{RMS}
Input full-scale at minimum gain (differential)	Gain=0dB		0.8*AVCC		V _{PP}
3dB bandwidth			20		kHz
Microphone mode	Input impedance		6	10	kΩ
Input impedance	Input capacitance			20	pF
THD+N (microphone input) @30mV _{RMS} input			0.02		%
THD+N (line input) @-3dBFS			0.01		%
ADC channels			2		

ADC channel variation	L vs. R variation		±0.2		dBFS
Analog supply voltage (AVCC)		1.70	2.8	3.0	V
MIC bias	AVCC=2.8V	AVCC*0.75		AVCC*0.9	V
	AVCC=1.7V	1.5V			
Digital supply voltage (DVDD)		0.99	1.1	1.21	V

17.5 Application and Implementation

17.5.1 Audio Output

Ameba-D audio codec supports three output modes: cap-less, differential and single-end. You can select a mode by setting the related registers. There are some differences in board circuit design among these modes.

17.5.1.1 Line-out Cap-less Mode

In this mode, the N-end of L/R channel outputs common-level voltage, while P-end drives the available analog audio signal. When earphone inserts into the jack, the ground must short with N-end output for audio signal de-couple. This is the reason that the ground is called virtual ground. The cap-less mode connection with a headphone jack is shown in Fig 17-2.

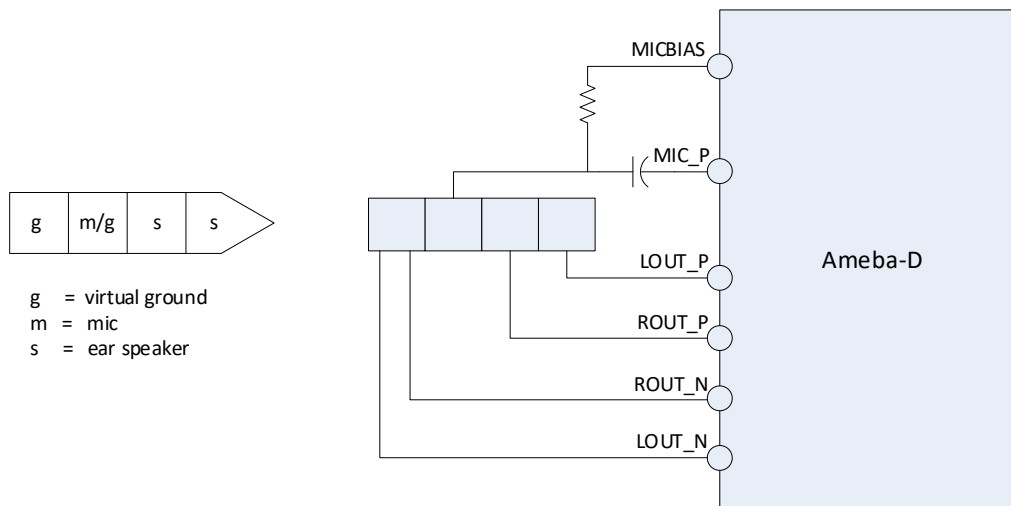


Fig 17-2 Cap-less mode connection with headphone jack

17.5.1.2 Line-out Differential Mode

In this mode, both N-end and P-end drive the available analog audio signal. Users should select the differential jack and earphone accordingly. The differential mode connection with a headphone jack is shown in Fig 17-3.

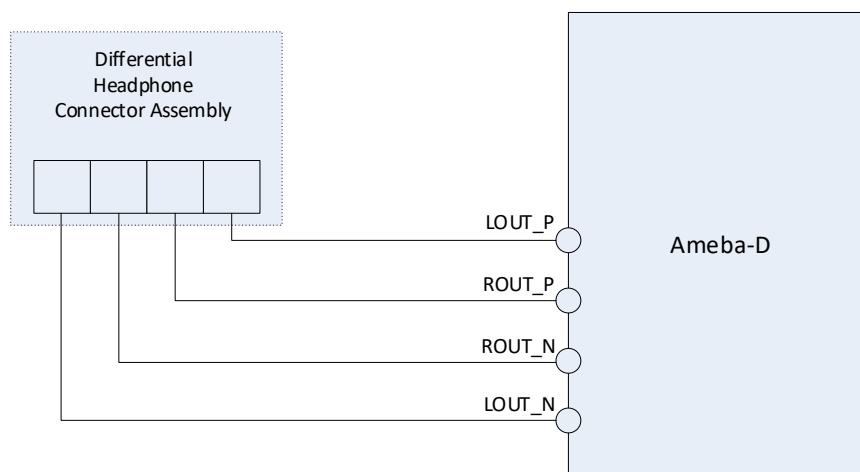


Fig 17-3 Differential mode connection with headphone jack

17.5.1.3 Line-out Single-end Mode

In this mode, board circuit designers need to place a capacitor to the P-end output path for analog audio signal pick-up. And no N-end output is required. The single-end mode connection with a headphone jack is shown in Fig 17-4.

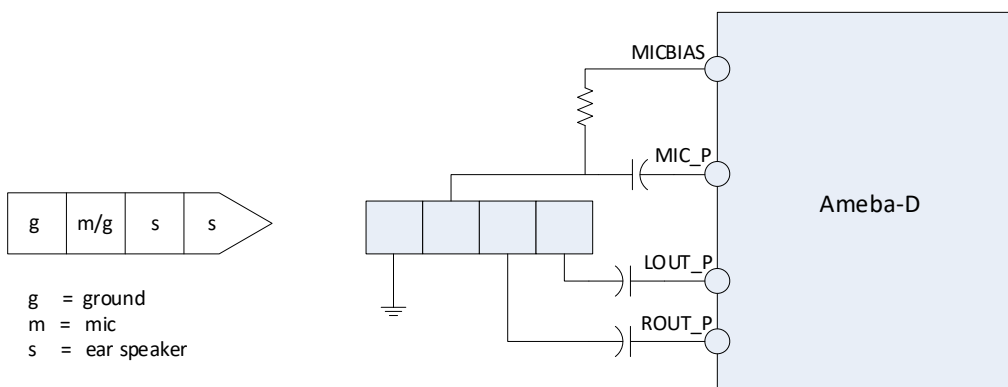


Fig 17-4 Single-end mode connection with headphone jack

17.5.2 Audio Input

The audio input path of Ameba-D audio codec includes line-in, single-end or differential analog mic and mono or stereo digital MIC. Users select the corresponding path according to project requirements.

17.5.2.1 Line-in

Fig 17-5 shows the situation of connecting the left channel of line-in signal to AUX_L, and connecting the right channel to AUX_R accordingly.

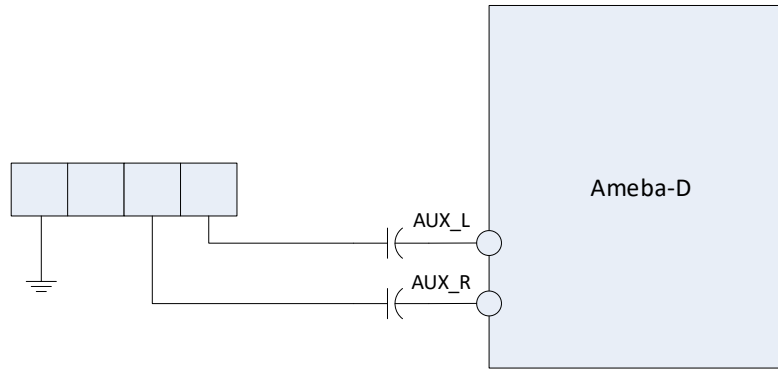


Fig 17-5 Line-in mode connection

17.5.2.2 Analog MIC

Fig 17-6 and Fig 17-7 show the situation of connecting MIC_P with single-end analog mic-phone or connecting MIC_P/MIC_N with differential analog mic-phone, while MICBIAS provides the mic-phone bias voltage.

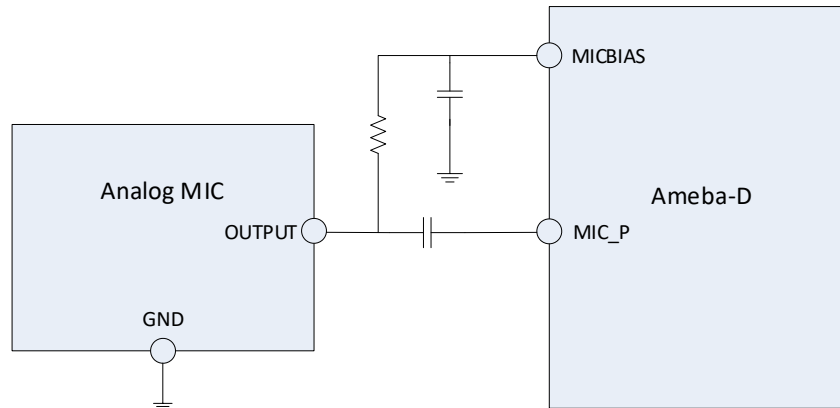


Fig 17-6 Analog MIC single-end mode connection

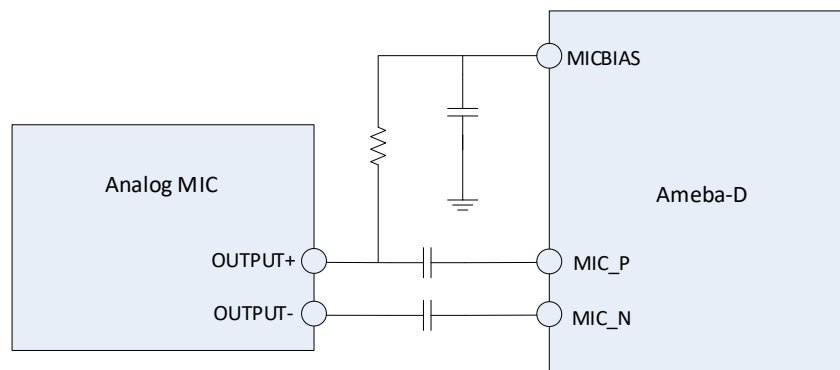


Fig 17-7 Analog MIC differential mode connection

17.5.2.3 Digital MIC

Fig 17-8 and Fig 17-9 show the situation of connecting DMIC_CLK/DMIC_DATA with digital mic-phone, while MICBIAS provides the mic-phone power supply.

- Tie the L/R of digital mic-phone to ground or VDD if only one digital mic-phone is placed.

- Tie the L/R of two digital mic-phones to ground and VDD respectively if a stereo mic-phone is needed. The two mic-phones will share the DMIC_DATA according to rising/falling edge.

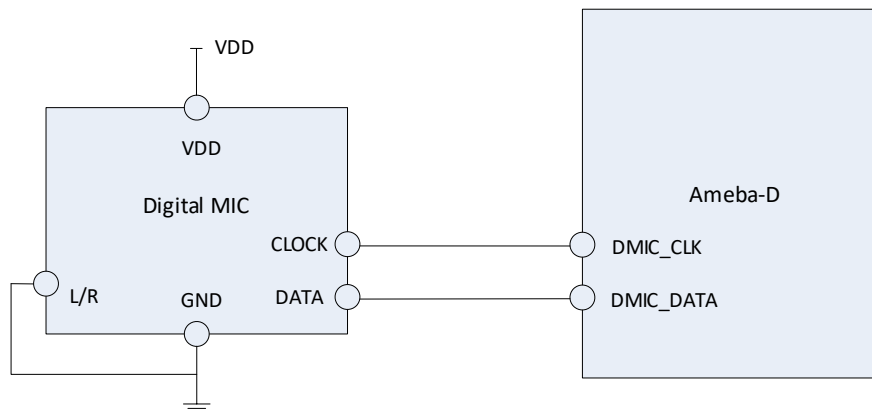


Fig 17-8 Digital MIC mono mode connection

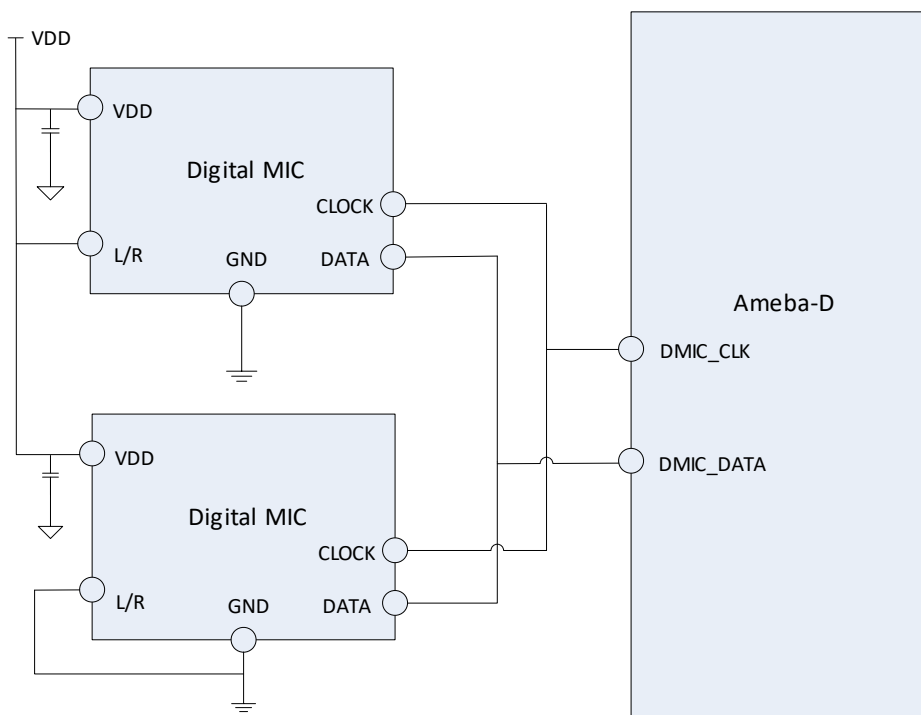


Fig 17-9 Digital MIC stereo mode connection

Fig 17-10 shows the mono PDM format, and the stereo PDM format is shown in Fig 17-11.

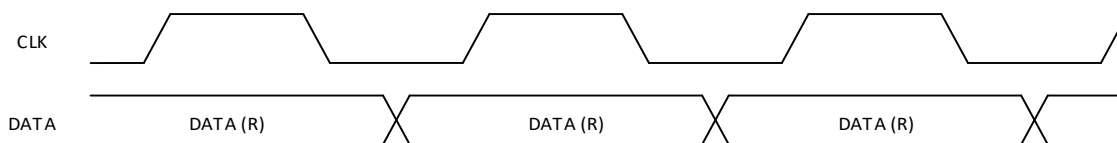


Fig 17-10 Mono PDM format

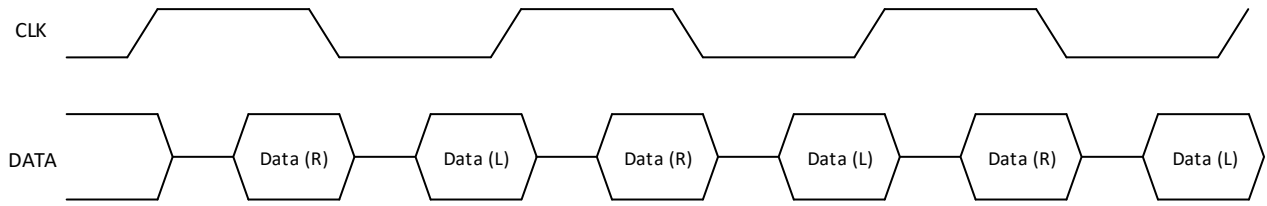


Fig 17-11 Stereo PDM format

Besides connecting digital mic-phone with DMIC_CLK/DMIC_DATA, you can also connect it with external I²S interface directly (see Fig 17-12).

- Set I²S bit clock to 64*Fs, that is 1MHz if Fs=16kHz.
- Need to implement decimation filter by software. It consumes about 4k cycles per 1024 bits on Cortex-M4, which means the computation complexity is about 4 MCPS if a 1MHz PDM clock is used.

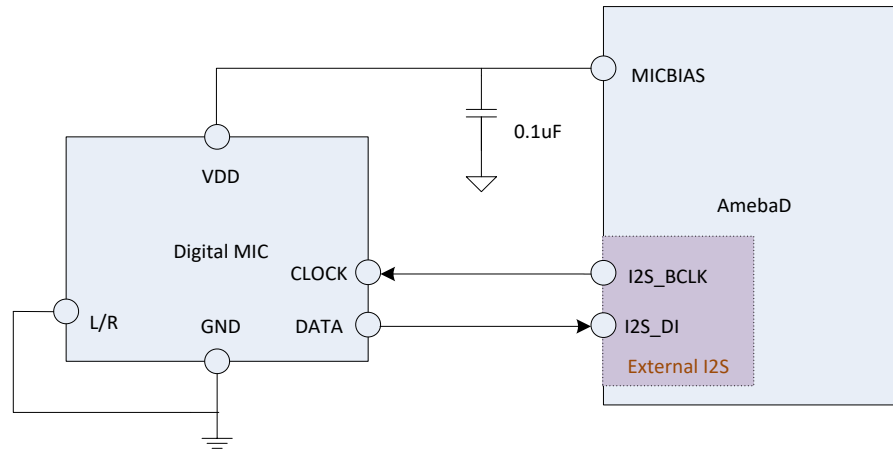


Fig 17-12 I²S acting as PDM

17.6 Registers

17.6.1 Analog Part

17.6.1.1 0x00

Address	Bit	Name	Access	Reset	Description
0x00	[0]	DAC_ADDACK_POW	R/W	1'b0	AD/DA clock power down control
	[1]	DAC_CKXEN	R/W	1'b0	DAC chopper clock enable control <ul style="list-style-type: none"> ● 1'b0: Disable ● 1'b1: Enable
	[2]	DAC_CKXSEL	R/W	1'b0	DAC chopper clock selection <ul style="list-style-type: none"> ● 1'b0: cks2/4 ● 1'b1: cks2/8
	[3]	DAC_L_POW	R/W	1'b0	DAC left channel power down control <ul style="list-style-type: none"> ● 1'b0: Power down ● 1'b1: Power on
	[4]	DAC_R_POW	R/W	1'b0	DAC right channel power down control <ul style="list-style-type: none"> ● 1'b0: Power down ● 1'b1: Power on
	[6:5]	DPRAMP_CSEL	R/W	2'b00	Depop C size selection <ul style="list-style-type: none"> ● 2'b00: 1x ● 2'b01: 2x

					<ul style="list-style-type: none"> 2'b10: 3x 2'b11: 4x
[7]	DPRAMP_ENRAMP	R/W	1'b0	DPRAMP enable ramp control	<ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
[8]	DPRAMP_POW	R/W	1'b0	DPRAMP power down control	<ul style="list-style-type: none"> 1'b0: Power down 1'b1: Power on
[9]	DTSDM_CKXEN	R/W	1'b0	ADC integrator 1 OP chopper enable	<ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
[10]	DTSDM_POW_L	R/W	1'b0	Left channel ADC power on control	<ul style="list-style-type: none"> 1'b0: Power down 1'b1: Power on
[11]	DTSDM_POW_R	R/W	1'b0	Right channel ADC power on control	<ul style="list-style-type: none"> 1'b0: Power down 1'b1: Power on
[12]	HPO_CLL	R/W	1'b0	Headphone left channel cap-less mode control	<ul style="list-style-type: none"> 1'b0: No cap-less 1'b1: Cap-less
[13]	HPO_CLNDPL	R/W	1'b0	Headphone left channel cap-less negative depop mode control	<ul style="list-style-type: none"> 1'b0: No depop 1'b1: Depop
[14]	HPO_CLNDPR	R/W	1'b0	Headphone right channel cap-less negative depop mode control	<ul style="list-style-type: none"> 1'b0: No depop 1'b1: Depop
[15]	HPO_CLPDPL	R/W	1'b0	Headphone left channel cap-less positive depop mode control	<ul style="list-style-type: none"> 1'b0: No depop 1'b1: Depop

17.6.1.2 0x01

Address	Bit	Name	Access	Reset	Description
0x01	[0]	HPO_CLPDPR	R/W	1'b0	Headphone right channel cap-less positive depop mode control <ul style="list-style-type: none"> 1'b0: No depop 1'b1: Depop
	[1]	HPO_CLR	R/W	1'b0	Headphone right channel cap-less mode control <ul style="list-style-type: none"> 1'b0: No cap-less 1'b1: Cap-less
	[3:2]	HPO DPRSELL	R/W	2'b00	Headphone left channel depop R size selection <ul style="list-style-type: none"> 2'b00: 1x 2'b01: 2x 2'b10: 3x 2'b11: 4x
	[5:4]	HPO DPRSELR	R/W	2'b00	Headphone right channel depop R size selection <ul style="list-style-type: none"> 2'b00: 1x 2'b01: 2x 2'b10: 3x 2'b11: 4x
	[6]	HPO_ENAL	R/W	1'b0	Headphone left channel enable amplifier control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[7]	HPO_ENAR	R/W	1'b0	Headphone right channel enable amplifier control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[8]	HPO_ENDPL	R/W	1'b0	Headphone left channel enable depop control

					<ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[9]	HPO_ENDPR	R/W	1'b0	Headphone right channel enable depop control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[10]	HPO_L_POW	R/W	1'b0	Headphone left channel power down control <ul style="list-style-type: none"> 1'b0: Power down 1'b1: Power on
	[11]	HPO_MDPL	R/W	1'b0	Headphone left channel mute depop mode control <ul style="list-style-type: none"> 1'b0: No depop 1'b1: Depop
	[12]	HPO_MDPR	R/W	1'b0	Headphone right channel mute depop mode control <ul style="list-style-type: none"> 1'b0: No depop 1'b1: Depop
	[13]	HPO_OPNDPL	R/W	1'b0	Headphone left channel op negative depop mode control <ul style="list-style-type: none"> 1'b0: No depop 1'b1: Depop
	[14]	HPO_OPNDPR	R/W	1'b0	Headphone right channel op negative depop mode control <ul style="list-style-type: none"> 1'b0: No depop 1'b1: Depop
	[15]	HPO_OPPDPL	R/W	1'b0	Headphone left channel op positive depop mode control <ul style="list-style-type: none"> 1'b0: No depop 1'b1: Depop

17.6.1.3 0x02

Address	Bit	Name	Access	Reset	Description
0x02	[1:0]	HPO_ML	R/W	2'b00	Headphone left channel mute control mute<0>: DAC mute<1>: Analog in <ul style="list-style-type: none"> 1'b0: Un-mute 1'b1: Mute
	[3:2]	HPO_MR	R/W	2'b00	Headphone right channel mute control mute<0>: DAC mute<1>: Analog in <ul style="list-style-type: none"> 1'b0: Un-mute 1'b1: Mute
	[4]	HPO_OPPDPR	R/W	1'b0	Headphone right channel op positive depop mode control <ul style="list-style-type: none"> 1'b0: No depop 1'b1: Depop
	[5]	HPO_R_POW	R/W	1'b0	Headphone right channel power down control <ul style="list-style-type: none"> 1'b0: Power down 1'b1: Power on
	[6]	HPO_SEL	R/W	1'b0	Headphone left channel single-end mode control <ul style="list-style-type: none"> 1'b0: Differential 1'b1: Single-end
	[7]	HPO_SER	R/W	1'b0	Headphone right channel single-end mode control <ul style="list-style-type: none"> 1'b0: Differential 1'b1: Single-end
	[8]	MBIAS_POW	R/W	1'b0	MBIAS power control <ul style="list-style-type: none"> 1'b0: Power down 1'b1: Power on
	[9]	MICBIAS_ENCHX	R/W	1'b0	MICBIAS enable chopper clock <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[10]	MICBIAS_POW	R/W	1'b0	MICBIAS power control

					<ul style="list-style-type: none"> 1'b0: Power down 1'b1: Power on
	[12:11]	MICBIAS_VSET	R/W	2'b00	MICBIAS select output voltage level <ul style="list-style-type: none"> 2'b00: 0.9*AVCC 2'b01: 0.86*AVCC (for AVCC=1.7v) 2'b10: 0.75*AVCC 2'b11: reserved
	[13]	MICBST_ENDFL	R/W	1'b0	MICBST left channel enable differential <ul style="list-style-type: none"> 1'b0: Single to differential 1'b1: Differential to differential
	[14]	MICBST_ENDFR	R/W	1'b0	MICBST right channel enable differential <ul style="list-style-type: none"> 1'b0: Single to differential 1'b1: Differential to differential
	[15]	VREF_POW	R/W	1'b0	VREF power control <ul style="list-style-type: none"> 1'b0: Power down 1'b1: Power on

17.6.1.4 0x03

Address	Bit	Name	Access	Reset	Description
0x03	[1:0]	MICBST_GSELL	R/W	2'b00	MICBST left channel gain select <ul style="list-style-type: none"> 2'b00: 0dB 2'b01: 20dB 2'b10: 30dB 2'b11: 40dB
	[3:2]	MICBST_GSELR	R/W	2'b00	MICBST right channel gain select <ul style="list-style-type: none"> 2'b00: 0dB 2'b01: 20dB 2'b10: 30dB 2'b11: 40dB
	[5:4]	MICBST_MUTE_L	R/W	2'b00	MICBST mute control mute<0>: MIC in mute<1>: Line in <ul style="list-style-type: none"> 1'b0: Un-mute 1'b1: Mute
	[7:6]	MICBST_MUTE_R	R/W	2'b00	MICBST mute control mute<0>: MIC in mute<1>: Line in <ul style="list-style-type: none"> 1'b0: Un-mute 1'b1: Mute
	[9:8]	MICBST_POW	R/W	2'b00	MICBST power control pow<0>: right channel pow<1>: left channel <ul style="list-style-type: none"> 1'b0: Power down 1'b1: Power on
	[11:10]	VREF_VREFSEL	R/W	2'b00	VREF voltage selection <ul style="list-style-type: none"> 2'b00: 0.52*VDD 2'b01: 0.51*VDD 2'b10: 0.50*VDD 2'b11: 0.49*VDD
	[15:12]	RSVD	N/A	2'b00	Reserved

17.6.1.5 0x04

Address	Bit	Name	Access	Reset	Description
0x04	[1:0]	DEBUG_BUS_SEL	R/W	2'b00	<ul style="list-style-type: none"> 2'b00: debug_bus_a

					<ul style="list-style-type: none"> 2'b01: debug_bus_b 2'b10: debug_bus_c 2'b11: debug_bus_d
	[15:2]	RSVD	N/A	14'h0	Reserved

17.6.2 Digital Part

17.6.2.1 0x0C

Address	Bit	Name	Access	Reset	Description
0x0C	[1:0]	SEL_BB_CK_DEPOP	R/W	2'b01	<ul style="list-style-type: none"> 2'b00: ck_depop=40MHz/1024 2'b01: ck_depop=40MHz/2048 2'b10: ck_depop=40MHz/4096 2'b11: ck_depop=40MHz/8192
	[2]	MICBIAS_OC	R	1'b0	the status flag of 'MICBIAS over-current protection'
	[3]	BB_CK_DEPOP_EN	R/W	1'b0	1'b1: enable 'ck_depop'
	[4]	CKX_MICBIAS_EN	R/W	1'b0	1'b1: enable 'ckx_micbias' to be 312.5kHz
	[15:5]	RSVD	N/A	11'h0	Reserved

17.6.2.2 0x0F

Address	Bit	Name	Access	Reset	Description
0x0F	[0]	SIDETONE_HPF_EN	R/W	1'b0	Sidetone processing enable control <ul style="list-style-type: none"> 1'b1: Enable sidetone HPF processing 1'b0: Disable sidetone HPF processing
	[3:1]	SIDETONE_HPF_FC_SEL	R/W	3'h0	Sidetone HPF cut-off frequency select (-6dB) <ul style="list-style-type: none"> 3'b000: fc = 120Hz 3'b001: fc = 239Hz 3'b010: fc = 358Hz 3'b011: fc = 477Hz 3'b100: fc = 597Hz 3'b101: fc = 716Hz 3'b110: fc = 835Hz 3'b111: fc = 955Hz
	[8:4]	SIDETONE_VOL_SEL	R/W	5'h0	Sidetone volume select <ul style="list-style-type: none"> 5'h00: -46.5dB 5'h01: -45.0dB 5'h02: -43.5dB 5'h03: -42.0dB 5'h04: -40.5dB 5'h05: -39.0dB 5'h06: -37.5dB 5'h07: -36.0dB 5'h08: -34.5dB 5'h09: -33.0dB 5'h0a: -31.5dB 5'h0b: -30.0dB 5'h0c: -28.5dB 5'h0d: -27.0dB 5'h0e: -25.5dB 5'h0f: -24.0dB 5'h10: -22.5dB 5'h11: -21.0dB 5'h12: -19.5dB 5'h13: -18.0dB

					<ul style="list-style-type: none"> 5'h14: -16.5dB 5'h15: -15.0dB 5'h16: -13.5dB 5'h17: -12.0dB 5'h18: -10.5dB 5'h19: -9.0dB 5'h1a: -7.5dB 5'h1b: -6.0dB 5'h1c: -4.5dB 5'h1d: -3.0dB 5'h1e: -1.5dB 5'h1f: 0dB
	[9]	SIDETONE_BOOST_SEL	R/W	1'b0	Sidetone boost select <ul style="list-style-type: none"> 1'b1: +12dB 1'b0: 0dB
	[11:10]	I2S_ADC_COMP	R/W	2'b00	<ul style="list-style-type: none"> 2'b00: OFF 2'b01: u law 2'b10: A law
	[13:12]	I2S_DAC_COMP	R/W	2'b00	<ul style="list-style-type: none"> 2'b00: OFF 2'b01: u law 2'b10: A law
	[15:14]	RSVD	N/A	2'b00	Reserved

17.6.2.3 0x10

Address	Bit	Name	Access	Reset	Description
0x10	[0]	EN_I2S_MONO	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: Mono 1'b0: Stereo
	[1]	I2S_EN_PCM_N_MODE	R/W	1'b0	<ul style="list-style-type: none"> 3'b000: I2S
	[3:2]	I2S_DATA_FORMAT_SEL	R/W	2'b00	<ul style="list-style-type: none"> 3'b010: Left Justified 3'b100: PCM mode A 3'b110: PCM mode B 3'b101: PCM mode A-N 3'b111: PCM mode B-N
	[5:4]	I2S_DATA_LEN_SEL	R/W	2'b00	<ul style="list-style-type: none"> 2'b00: 16 bits 2'b10: 24 bits 2'b11: 8 bits
	[6]	INV_I2S_SCLK	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: I2S/PCM bit clock is inverted
	[7]	I2S_RST_N_REG	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: rst_n to audio digital IP is de-asserted 1'b0: rst_n to audio digital IP is asserted
	[9:8]	SEL_I2S_RX_CH	R/W	2'b00	<ul style="list-style-type: none"> 2'b00: L/R 2'b01: R/L 2'b10: L/L 2'b11: R/R @ ADC path
	[11:10]	SEL_I2S_TX_CH	R/W	2'b00	<ul style="list-style-type: none"> 2'b00: L/R 2'b01: R/L 2'b10: L/L 2'b11: R/R @ DAC path
	[12]	STEREO_I2S_SELF_LPBK_EN	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: Internal loopback mode is enabled
	[15:13]	RSVD	N/A	3'h0	Reserved

17.6.2.4 0x11

Address	Bit	Name	Access	Reset	Description
0x11	[0]	ADC_L_DMIC_RI_FA_SEL	R/W	1'b0	DMIC Data Latching Control <ul style="list-style-type: none"> 1'b0: Rising latch

					<ul style="list-style-type: none"> 1'b1: Falling latch
[2:1]	RSVD	N/A	2'b00	Reserved	
[3]	ADC_L_DMIC_LPF2ND_EN	R/W	1'b1	DMIC SRC 2nd LPF control	<ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
[4]	ADC_L_DMIC_LPF1ST_EN	R/W	1'b1	DMIC SRC 1st LPF control	<ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
[6:5]	ADC_L_DMIC_LPF1ST_FC_SEL	R/W	2'b00	DMIC SRC 1st LPF fc	<ul style="list-style-type: none"> 2'b00: 31.04khz 2'b01: 46.92khz 2'b10: 63.06khz 2'b11: 79.45khz
[7]	RSVD	N/A	1'b0	Reserved	
[8]	ADC_L_AD_LPF2ND_EN	R/W	1'b1	ADC SRC 2nd LPF control	<ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
[9]	ADC_L_AD_LPF1ST_EN	R/W	1'b1	ADC SRC 1st LPF control	<ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
[11:10]	ADC_L_AD_LPF1ST_FC_SEL	R/W	2'b00	ADC SRC 1st LPF fc	<ul style="list-style-type: none"> 2'b00: 31.04khz 2'b01: 46.92khz 2'b10: 63.06khz 2'b11: 79.45khz
[12]	ADC_L_AD_MIX_MUTE	R/W	1'b1	analog ADC input path mute control Left Channel	<ul style="list-style-type: none"> 1'b0: Un-Mute 1'b1: Mute
[13]	ADC_L_DMIC_MIX_MUTE	R/W	1'b1	DMIC input path mute control Left Channel	<ul style="list-style-type: none"> 1'b0: Un-Mute 1'b1: Mute
[14]	ADC_L_AD_DCHPF_EN	R/W	1'b1	High pass filter enable control (filter DC)	<ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
[15]	RSVD	N/A	1'b0	Reserved	

17.6.2.5 0x12

Address	Bit	Name	Access	Reset	Description
0x12	[1:0]	ADC_L_AD_COMP_GAIN	R/W	2'b00	ADC compensate gain <ul style="list-style-type: none"> 2'b00: 0dB 2'b01: 1dB 2'b10: 2dB 2'b11: 3dB
	[2]	ADC_L_ADJ_HPF_2ND_EN	R/W	1'b1	Adaptive 2nd High pass filter enable control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[5:3]	ADC_L_ADJ_HPF_COEF_SEL	R/W	3'b000	Coefficient coarse select, fc range (num==0 ~ num==63) <ul style="list-style-type: none"> 3'b000: fs = 8k, or 16k corresponding fc = (20~2000Hz)/(40~4000Hz) 3'b001: fs = 32k corresponding fc = (40~3278Hz) 3'b010: fs = 48k or 44.1k corresponding fc = (30~2168Hz)/(28~1992Hz) 3'b011: fs = 96k or 88.2k corresponding fc = (30~2034Hz)/(28~1869Hz)
	[7:6]	ADC_L_DMIC_BOOST_GAIN	R/W	2'b00	DMIC boost gain control

					<ul style="list-style-type: none"> 2'b00: 0dB 2'b01: 12dB 2'b10: 24dB 2'b11: 36dB
	[8]	ADC_L_AD_MUTE	R/W	1'b0	Digital Mute from ADC Left Channel Digital Mixer <ul style="list-style-type: none"> 1'b0: Un-Mute 1'b1: Mute
	[10:9]	ADC_L_AD_ZDET_FUNC	R/W	2'b10	ADC zero detection function select <ul style="list-style-type: none"> 2'b00: Immediate change 2'b01: Zero detection & immediate change 2'b10: Zero detection & increase/decrease change 2'b11: Un-change
	[12:11]	ADC_L_AD_ZDET_TOUT	R/W	2'b01	ADC zero detection time out select <ul style="list-style-type: none"> 2'b00: 1024*16 samples 2'b01: 1024*32 samples 2'b10: 1024*64 samples 2'b11: 64 samples
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.2.6 0x13

Address	Bit	Name	Access	Reset	Description
0x13	[5:0]	ADC_L_ADJ_HPF_COEF_NUM	R/W	6'h00	coefficient fine select (0~63)
	[12:6]	ADC_L_AD_GAIN	R/W	7'h2F	ADC digital volume -17.625dB~+30dB in 0.375dB step <ul style="list-style-type: none"> 7'h00: -17.625dB ... 7'h2f: 0dB 7'h30: 0.375dB ... 7'h7f: 30dB
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.2.7 0x14

Address	Bit	Name	Access	Reset	Description
0x14	[0]	ADC_R_DMIC_RI_FA_SEL	R/W	1'b0	DMIC Data Latching Control <ul style="list-style-type: none"> 1'b0: Rising latch 1'b1: Falling latch
	[2:1]	RSVD	N/A	2'b00	Reserved
	[3]	ADC_R_DMIC_LPF2ND_EN	R/W	1'b1	DMIC SRC 2nd LPF control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[4]	ADC_R_DMIC_LPF1ST_EN	R/W	1'b1	DMIC SRC 1st LPF control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[6:5]	ADC_R_DMIC_LPF1ST_FC_SEL	R/W	2'b00	DMIC SRC 1st LPF fc <ul style="list-style-type: none"> 2'b00: 31.04khz 2'b01: 46.92khz 2'b10: 63.06khz 2'b11: 79.45khz
	[7]	RSVD	N/A	1'b0	Reserved
	[8]	ADC_R_AD_LPF2ND_EN	R/W	1'b1	ADC SRC 2nd LPF control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[9]	ADC_R_AD_LPF1ST_EN	R/W	1'b1	ADC SRC 1st LPF control

					<ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[11:10]	ADC_R_AD_LPF1ST_FC_SEL	R/W	2'b00	ADC SRC 1st LPF fc <ul style="list-style-type: none"> 2'b00: 31.04khz 2'b01: 46.92khz 2'b10: 63.06khz 2'b11: 79.45khz
	[12]	ADC_R_AD_MIX_MUTE	R/W	1'b1	analog ADC input path mute control Left Channel <ul style="list-style-type: none"> 1'b0: Un-Mute 1'b1: Mute
	[13]	ADC_R_DMIC_MIX_MUTE	R/W	1'b1	DMIC input path mute control Left Channel <ul style="list-style-type: none"> 1'b0: Un-Mute 1'b1: Mute
	[14]	ADC_R_AD_DCHPF_EN	R/W	1'b1	High pass filter enable control (filter DC) <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[15]	RSVD	N/A	1'b0	Reserved

17.6.2.8 0x15

Address	Bit	Name	Access	Reset	Description
0x15	[1:0]	ADC_R_AD_COMP_GAIN	R/W	2'b00	ADC compensate gain <ul style="list-style-type: none"> 2'b00: 0dB 2'b01: 1dB 2'b10: 2dB 2'b11: 3dB
	[2]	ADC_R_ADJ_HPF_2ND_EN	R/W	1'b1	Adaptive 2nd High pass filter enable control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[5:3]	ADC_R_ADJ_HPF_COEF_SEL	R/W	3'b000	Coefficient coarse select, fc range (num==0 ~ num==63) <ul style="list-style-type: none"> 3'b000: fs = 8k, or 16k corresponding fc = (20~2000Hz)/(40~4000Hz) 3'b001: fs = 32k corresponding fc = (40~3278Hz) 3'b010: fs = 48k or 44.1k corresponding fc = (30~2168Hz)/(28~1992Hz) 3'b011: fs = 96k or 88.2k corresponding fc = (30~2034Hz)/(28~1869Hz)
	[7:6]	ADC_R_DMIC_BOOST_GAIN	R/W	2'b00	DMIC boost gain control <ul style="list-style-type: none"> 2'b00: 0dB 2'b01: 12dB 2'b10: 24dB 2'b11: 36dB
	[8]	ADC_R_AD_MUTE	R/W	1'b0	Digital Mute from ADC Left Channel Digital Mixer <ul style="list-style-type: none"> 1'b0: Un-Mute 1'b1: Mute
	[10:9]	ADC_R_AD_ZDET_FUNC	R/W	2'b10	ADC zero detection function select <ul style="list-style-type: none"> 2'b00: Immediate change 2'b01: Zero detection & immediate change 2'b10: Zero detection & increase/decrease change 2'b11: Unchange
	[12:11]	ADC_R_AD_ZDET_TOUT	R/W	2'b01	ADC zero detection time out select <ul style="list-style-type: none"> 2'b00: 1024*16 samples 2'b01: 1024*32 samples 2'b10: 1024*64 samples 2'b11: 64 samples
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.2.9 0x16

Address	Bit	Name	Access	Reset	Description
0x16	[5:0]	ADC_R_ADJ_HPF_COEF_NUM	R/W	6'h00	Coefficient fine select (0~63)
	[12:6]	ADC_R_AD_GAIN	R/W	7'h2F	ADC digital volume -17.625dB~+30dB in 0.375dB step <ul style="list-style-type: none"> ● 7'h00: -17.625dB ● ... ● 7'h2f: 0dB ● 7'h30: 0.375dB ● ... ● 7'h7f: 30dB
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.2.10 0x17

Address	Bit	Name	Access	Reset	Description
0x17	[3:0]	DAC_SAMPLE_RATE	R/W	4'h0	Set DAC sample rate <ul style="list-style-type: none"> ● 4'h0: 48K ● 4'h1: 96K ● 4'h2: Reserved ● 4'h3: 32K ● 4'h4: Reserved ● 4'h5: 16K ● 4'h6: Reserved ● 4'h7: 8K ● 4'h8: 44.1K ● 4'h9: 88.2K ● 4'hA~4'hF: Reserved
	[7:4]	ADC_SAMPLE_RATE	R/W	4'h0	Set ADC sample rate <ul style="list-style-type: none"> ● 4'h0: 48K ● 4'h1: 96K ● 4'h2: Reserved ● 4'h3: 32K ● 4'h4: Reserved ● 4'h5: 16K ● 4'h6: Reserved ● 4'h7: 8K ● 4'h8: 44.1K ● 4'h9: 88.2K ● 4'hA~4'hF: Reserved
	[10:8]	DMIC_CLK_SEL	R/W	3'h1	Set clock of digital microphone <ul style="list-style-type: none"> ● 3'b000: 5MHz ● 3'b001: 2.5MHz ● 3'b010: 1.25MHz ● 3'b011: 625kHz ● 3'b100: 312.5kHz
	[14:11]	ASRC_FSI_RATE_MANUAL	R/W	4'h0	Set input sample rate by user <ul style="list-style-type: none"> ● 4'h0: 48K ● 4'h1: 96K ● 4'h2: 192K ● 4'h3: 32K ● 4'h4: Reserved ● 4'h5: 16K ● 4'h6: Reserved ● 4'h7: 8K ● 4'h8: 44.1K

					<ul style="list-style-type: none"> ● 4'h9: 88.2K ● 4'hA: 176.4K ● 4'hB~4'hF: Reserved
	[15]	ASRC_FSI_GATING_EN	R/W	1'b0	Set FSI gating <ul style="list-style-type: none"> ● 1'b0: Disable ● 1'b1: Enable FSI gating

17.6.2.11 0x18

Address	Bit	Name	Access	Reset	Description
0x18	[0]	DA_L_EN	R/W	1'b0	Set DAC filter left channel clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[1]	DA_R_EN	R/W	1'b0	Set DAC filter right channel clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[2]	MOD_L_EN	R/W	1'b0	Set DAC modulation left channel clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[3]	MOD_R_EN	R/W	1'b0	Set DAC modulation right channel clock <ul style="list-style-type: none"> ● 1'b0: TURN off clock and reset ● 1'b1: turn on clock
	[4]	DA_ANA_CLK_EN	R/W	1'b0	Set DAC analog clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock ● 1'b1: Turn on clock
	[5]	DA_FIFO_EN	R/W	1'b0	Set DAC FIFO clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[6]	ST_EN	R/W	1'b0	Set sidetone clock <ul style="list-style-type: none"> ● 1'b0: TURN off clock and reset ● 1'b1: turn on clock
	[7]	AD_L_EN	R/W	1'b0	Set ADC filter left channel clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[8]	AD_R_EN	R/W	1'b0	Set ADC filter right channel clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[9]	AD_FIFO_EN	R/W	1'b0	Set ADC FIFO clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[10]	AD_ANA_CLK_EN	R/W	1'b0	Set ADC analog clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock ● 1'b1: Turn on clock
	[11]	AD_ANA_L_EN	R/W	1'b0	Set ADC filter left channel analog ADC path clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[12]	AD_ANA_R_EN	R/W	1'b0	Set ADC filter right channel analog ADC path clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[13]	DMIC_L_EN	R/W	1'b0	Set ADC filter left channel digital MIC path clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[14]	DMIC_R_EN	R/W	1'b0	Set ADC filter right channel digital MIC path clock <ul style="list-style-type: none"> ● 1'b0: Turn off clock and reset ● 1'b1: Turn on clock
	[15]	DMIC_CLK_EN	R/W	1'b1	Set digital MIC clock

					<ul style="list-style-type: none"> 1'b0: Turn off clock 1'b1: Turn on clock
--	--	--	--	--	---

17.6.2.12 0x19

Address	Bit	Name	Access	Reset	Description
0x19	[15:0]	ASRC_FTK_SDM_INI[15:0]	R/W	16'h0	Set initial value of tracked frequency

17.6.2.13 0x1A

Address	Bit	Name	Access	Reset	Description
0x1A	[7:0]	ASRC_FTK_SDM_INI[23:16]	R/W	8'h0	Set initial value of tracked frequency
	[8]	GEN_SRC_16K128_EN	R/W	1'b1	Set 16k*128 clock <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[9]	GEN_SRC_32K128_EN	R/W	1'b1	Set 32k*128 clock <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[10]	GEN_SRC_44P1K128_EN	R/W	1'b1	Set 44.1k*128 clock <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[11]	GEN_SRC_48K128_EN	R/W	1'b1	Set 48k*128 clock <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[12]	GEN_SRC_8K128_EN	R/W	1'b1	Set 8k*128 clock <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[13]	AD_ANA_CLK_SEL	R/W	1'b0	Set clk_ad_ana phase (reference clk_da_ana) <ul style="list-style-type: none"> 1'b0: Inphase 1'b1: Inverse phase
	[15:14]	ASRC_FTK_LOOP_GAIN_SEL	R/W	2'b01	Set loop gain <ul style="list-style-type: none"> 2'b00: 2⁻⁸ 2'b01: 2⁻¹⁴ 2'b10: 2⁻¹⁸ 2'b11: 2⁻²⁰

17.6.2.14 0x1B

Address	Bit	Name	Access	Reset	Description
0x1B	[0]	AUDIO_IP_TCON_EN	R/W	1'b1	Set audio ip tcon <ul style="list-style-type: none"> 1'b0: Disable and reset 1'b1: Enable
	[1]	ASRC_FTK_LOOP_EN	R/W	1'b0	Set loop filter enable <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[3:2]	ASRC_256FS_SYS_SEL	R/W	2'h0	Set clock of src_tcon <ul style="list-style-type: none"> 2'b00: 512*48K 2'b01: 1024*48K 2'b10: 2048*48K 2'b11: Reserved
	[4]	ASRC_EN	R/W	1'b0	Set asynchronous sample rate conversion <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[6:5]	SIDETONE_IN_SEL	R/W	2'h0	sidetone input selection <ul style="list-style-type: none"> 2'b00: From adc_l_lpf

					<ul style="list-style-type: none"> ● 2'b01: From adc_r_lpf ● 2'b10: From dmic_l_lpf ● 2'b11: From dmic_r_lpf
	[15:7]	RSVD	N/A	9'h0	Reserved

17.6.2.15 0x1C

Address	Bit	Name	Access	Reset	Description
0x1C	[0]	ALC_ATK_MODE_RC_CTR	R/W	1'b0	ALC force slow recovery enable <ul style="list-style-type: none"> ● 1'b0: Force slow recovery ● 1'b1: Depend on alc_rc_mode_l and alc_rc_mode_r
	[5:1]	ALC_ATK_NOISE_RATE	R/W	5'h06	ALC attack time in noise gate mode attack time = $(4 * 2^n)/48k$, $n = \text{alc_atk_noise_rate}[4:0]$ <ul style="list-style-type: none"> ● 5'h00: 83us ● 5'h01: 166us ● 5'h02: 332us ● ... ● 5'h12: 21.85s
	[10:6]	ALC_ATK_RATE	R/W	5'h03	ALC attack time attack time = $(4 * 2^n)/48k$, $n = \text{alc_atk_rate}[4:0]$ <ul style="list-style-type: none"> ● 5'h00: 83us ● 5'h01: 166us ● 5'h02: 332us ● ... ● 5'h12: 21.85s
	[15:11]	ALC_ATK_SPEED_UP_RATE	R/W	5'h0	ALC attack time for speed up mode attack time = $(2 * 2^n)/48k$, $n = \text{alc_atk_rate}[4:0]$ <ul style="list-style-type: none"> ● 5'h00: 42us ● 5'h01: 83us ● 5'h02: 166us ● ... ● 5'h13: 21.85s

17.6.2.16 0x1D

Address	Bit	Name	Access	Reset	Description
0x1D	[0]	ALC_SPEED_UP_EN	R/W	1'b0	alc_speed_up function enable control <ul style="list-style-type: none"> ● 1'b0: Disable alc_speed_up function ● 1'b1: Enable alc_speed_up function
	[1]	ALC_ZD_FULL_OV_EN	R/W	1'b0	enable the controlling of zero-cross by full-scale flag <ul style="list-style-type: none"> ● 1'b0: Zero-cross function is controlled by register ● 1'b1: Zero-cross function is controlled by full-scale flag
	[8:2]	ALC_BK_GAIN_L	R/W	7'h5F	ALC back boost gain control (0.375dB/step) <ul style="list-style-type: none"> ● 7'h00: -35.625dB ● 7'h01: -35.250dB ● ... ● 7'h5f: 0.000dB ● ... ● 7'h7f: 12.000dB
	[15:9]	ALC_BK_GAIN_R	R/W	7'h5F	ALC back boost gain control (0.375dB/step) <ul style="list-style-type: none"> ● 7'h00: -35.625dB ● 7'h01: -35.250dB ● ... ● 7'h5f: 0.000dB ● ... ● 7'h7f: 12.000dB

17.6.2.17 0x1E

Address	Bit	Name	Access	Reset	Description
0x1E	[0]	ALC_CTR_EN	R/W	1'b0	ALC function & volume control <ul style="list-style-type: none"> 1'b0: Function OFF but volume ON 1'b1: Function ON and volume ON
	[1]	ALC_DRC_EN	R/W	1'b0	DRC enable control <ul style="list-style-type: none"> 1'b0: DISABLE DRC 1'b1: enable DRC
	[3:2]	ALC_DRC_RATIO_SEL	R/W	2'b01	ALC DRC compression ratio select control <ul style="list-style-type: none"> 2'b00: 1:1 2'b01: 1:2 2'b10: 1:4 2'b11: 1:8
	[5:4]	ALC_DRC_RATIO_SEL2	R/W	2'b01	ALC DRC compression ratio select control <ul style="list-style-type: none"> 2'b00: 1:1 2'b01: 1:2 2'b10: 1:4 2'b11: 1:8
	[6]	ALC_EN	R/W	1'b0	ALC enable control <ul style="list-style-type: none"> 1'b0: Disable ALC 1'b1: Enable ALC
	[7]	ALC_FORCE_FAST_RC_CTR	R/W	1'b0	ALC force fast recovery control <ul style="list-style-type: none"> 1'b0: Disable force fast recovery 1'b1: Enable force fast recovery
	[8]	ALC_FORCE_FAST_RC_EN	R/W	1'b1	ALC force fast recovery control for special cases (zero data mode) <ul style="list-style-type: none"> 1'b0: Disable force fast recovery for zero data mode 1'b1: Enable force fast recovery for zero data mode
	[14:9]	ALC_FT_BOOST	R/W	6'h0	ALC front boost gain control (0.75dB/step) <ul style="list-style-type: none"> 6'h00: 0dB 6'h01: 0.75dB ... 6'h27: 29.25dB
	[15]	RSVD	N/A	1'b0	Reserved

17.6.2.18 0x1F

Address	Bit	Name	Access	Reset	Description
0x1F	[2:0]	ALC_LPF_COEF_SEL	R/W	3'h3	Coefficient for low pass filter in ALC <ul style="list-style-type: none"> 3'b000: 2⁻³ 3'b001: 2⁻⁴ 3'b010: 2⁻⁵ 3'b011: 2⁻⁶ 3'b100: 2⁻⁷ 3'b101: 2⁻⁸ 3'b110: 2⁻¹⁰ 3'b111: 2⁻¹¹
	[5:3]	ALC_MIN_RANGE	R/W	3'h2	Threshold margin control in ALC <ul style="list-style-type: none"> 3'b000: +-0dB 3'b001: +-0.375dB 3'b010: +-0.75dB 3'b011: +-1.125dB 3'b100: +-1.5dB 3'b101: +-1.875dB 3'b110: +-2.25dB 3'b111: +-2.625dB

	[6]	ALC_NOISE_GAIN_HD	R/W	1'b0	ALC hold gain when noise gate mode <ul style="list-style-type: none"> 1'b0: Don't hold gain in noise gate mode (depending on noise gate threshold to do noise gate's AGC) 1'b1: Hold gain in noise gate mode
	[10:7]	ALC_NOISE_GATE_EXP	R/W	4'h4	ALC noise_gate boost control <ul style="list-style-type: none"> 4'b0000: 0dB 4'b0001: 3dB 4'b0010: 6dB ... 4'b1111: 45dB (3dB/step)
	[11]	ALC_NOISE_GATE_EN	R/W	1'b0	ALC noise_gate mode control <ul style="list-style-type: none"> 1'b0: Disable noise_gate mode 1'b1: Enable noise_gate mode
	[12]	ALC_NOISE_GATE_DROP_EN	R/W	1'b0	ALC noise_gate drop mode control <ul style="list-style-type: none"> 1'b0: Disable noise_gate_drop mode 1'b1: Enable noise_gate_drop mode
	[14:13]	ALC_NOISE_GATE_RATIO_SEL	R/W	2'b01	ALC DRC expansion ratio select control, when noise gate is enabled <ul style="list-style-type: none"> 2'b00: 1:1 2'b01: 2:1 2'b10: 4:1 2'b11: 8:1
	[15]	RSVD	N/A	1'b0	Reserved

17.6.2.19 0x20

Address	Bit	Name	Access	Reset	Description
0x20	[1:0]	ALC_NOISE_RANGE	R/W	2'h1	Noise gate threshold margin control in ALC <ul style="list-style-type: none"> 2'b00: +-0dB 2'b01: +-1.5dB 2'b10: +-3.0dB 2'b11: +-4.5dB
	[9:2]	ALC_OFF_GAIN	R/W	8'hFF	ALC digital volume gain when alc_ctr_en=1'b0 <ul style="list-style-type: none"> 8'hff: 0dB 8'hfe: -0.375dB 8'hfd: -0.75dB ... 8'h00: -95.625dB (-0.375dB/step)
	[12:10]	ALC_RATE_SEL	R/W	3'h1	ALC rate control for sample rate change <ul style="list-style-type: none"> 3'b001: 48kHz 3'b010: 96kHz 3'b011: 192kHz 3'b101: 44.1kHz 3'b100: 88.2kHz 3'b101: 176.4kHz
	[13]	ALC_RC_FAST_EN	R/W	1'b0	ALC force fast recovery control for special cases (window) <ul style="list-style-type: none"> 1'b0: Disable force fast recovery for special cases 1'b1: Enable force fast recovery for normal use
	[15:14]	ALC_COMP_RATIO_SEL	R/W	2'b11	ALC DRC compression ratio select control <ul style="list-style-type: none"> 2'b00: 1:1 2'b01: 1:2 2'b10: 1:4 2'b11: 1:8

17.6.2.20 0x21

Address	Bit	Name	Access	Reset	Description
0x21	[4:0]	ALC_RC_FAST_RATE	R/W	5'h4	ALC fast recovery time recovery time = $(4 * 2^n)/48k$, $n = alc_rc_fast_rate[4:0]$ <ul style="list-style-type: none"> 5'h00: 83us 5'h01: 166us 5'h02: 332us ... 5'h12: 21.85s
	[9:5]	ALC_RC_SLOW_RATE	R/W	5'h0C	ALC fast recovery time recovery time = $(4 * 2^n)/48k$, $n = alc_rc_slow_rate[4:0]$ <ul style="list-style-type: none"> 5'h00: 83us 5'h01: 166us 5'h02: 332us ... 5'h12: 21.85s
	[15:10]	ALC_THMAX	R/W	6'h0	ALC main-limiter threshold level (at amplitude domain) control <ul style="list-style-type: none"> 6'h00: 0dB 6'h01: -0.375dB 6'h02: -0.750dB ... 6'h3f: -23.625dB (-0.375dB/step)

17.6.2.21 0x22

Address	Bit	Name	Access	Reset	Description
0x22	[7:0]	ALC_RC_WD_MAX	R/W	8'h74	Set upper bound of fast recovery window <ul style="list-style-type: none"> 8'h01: 5.33ms 8'h02: 10.67ms ... 8'hff: 1360ms
	[15:8]	ALC_RC_WD_MIN	R/W	8'h18	Set lower bound of fast recovery window <ul style="list-style-type: none"> 8'h01: 2.67ms 8'h02: 5.33ms ... 8'hff: 680ms

17.6.2.22 0x23

Address	Bit	Name	Access	Reset	Description
0x23	[1:0]	ALC_THFULL	R/W	2'h0	ALC full scale threshold level (at amplitude domain) control <ul style="list-style-type: none"> 2'b00: 0dBfs 2'b01: -0.5dBfs 2'b10: -1.0dBfs 2'b11: -1.5dBfs
	[7:2]	ALC_THMAX2	R/W	6'h0	ALC second-limiter threshold level (at amplitude domain) control <ul style="list-style-type: none"> 6'h00: 0dB 6'h01: -0.75dB 6'h02: -1.5dB ... 6'h3c: -45dB (-0.75dB/step)
	[12:8]	ALC_THNOISE	R/W	5'h0	AGC noise gate threshold level (at amplitude domain) control <ul style="list-style-type: none"> 5'h00: -24.00dB 5'h01: -25.50dB

					<ul style="list-style-type: none"> ● 5'h02: -27.00dB ● ... ● 5'h1f: -70.50dB (-1.5dB/step)
	[15:13]	ALC_THZERO	R/W	3'h0	ALC zero-data threshold level (at amplitude domain) control <ul style="list-style-type: none"> ● 3'b000: -84.0dB ● 3'b001: -85.5dB ● 3'b010: -87.0dB ● ... ● 3'b111: -94.5dB (-1.5dB/step)

17.6.2.23 0x24

Address	Bit	Name	Access	Reset	Description
0x24	[1:0]	ALC_ZDET_FUNC	R/W	2'h1	Volume control for ALC zero detection <ul style="list-style-type: none"> ● 2'b00: immediately change ● 2'b01: change directly when zero crossing ● 2'b10: change per step when zero crossing ● 2'b11: hold original gain
	[3:2]	ALC_ZDET_TOUT	R/W	2'h1	Time out for volume change with ALC zero detection <ul style="list-style-type: none"> ● 2'b00: 12000*SYNC ● 2'b01: 24000*SYNC ● 2'b10: 36000*SYNC ● 2'b11: 48000*SYNC
	[4]	ALC_ZERO_DATA_EN	R/W	1'b1	ALC zero_data mode control <ul style="list-style-type: none"> ● 1'b0: Disable zero_data mode ● 1'b1: Enable zero_data mode
	[7:5]	ALC_LIMITER_RATIO	R/W	3'h7	ALC limiter ratio control <ul style="list-style-type: none"> ● 3'b000: 1/4 ● 3'b001: 1/8 ● 3'b010: 1/16 ● 3'b011: 1/32 ● 3'b100: 1/64 ● 3'b101: 1/128 ● 3'b110: 1/256 ● 3'b111: hard limiter
	[8]	ALC_ZERO_DATA_SEL	R/W	1'b1	ALC zero data mode selection control <ul style="list-style-type: none"> ● 1'b0: Decide zero data mode by root mean square ● 1'b1: Decide zero data mode by amplitude detection
	[10:9]	ALC_ZERO_DATA_CNT_TH	R/W	2'h2	Time out for ALC zero data mode with amplitude detection <ul style="list-style-type: none"> ● 2'b00: 1024 samples (20ms with 48k sample rate) ● 2'b01: 2048 samples (40ms with 48k sample rate) ● 2'b10: 4096 samples (80ms with 48k sample rate) ● 2'b11: 8 samples (0.17ms with 48k sample rate, test mode)
	[13:11]	ALC_ZERO_DATA_LSB_SEL	R/W	3'h1	Threshold for ALC zero data mode with amplitude detection <ul style="list-style-type: none"> ● 3'b000: -78db ● 3'b001: -84db ● 3'b010: -90db ● 3'b011: -102db ● 3'b100: -108db ● 3'b101: -114db ● 3'b110: -126db ● 3'b111: -132db
	[14]	ALC_BYPASS_CD_EN	R/W	1'b1	cross-detection control for alc_bypass function <ul style="list-style-type: none"> ● 1'b0: alc_bypass function without cross detection ● 1'b1: alc_bypass function with cross detection
	[15]	ALC_ATK_HOLD_EN	R/W	1'b0	enable control for attack hold function in ALC <ul style="list-style-type: none"> ● 1'b0: Attack hold function is disabled

					<ul style="list-style-type: none"> 1'b1: Attack hold function is enabled
--	--	--	--	--	---

17.6.2.24 0x25 ~ 0x26

Address	Bit	Name	Access	Reset	Description
0x25	[15:0]	ALC_LIMITER_TH[15:0]	R/W	16'h0	ALC limiter threshold control
0x26	[7:0]	ALC_LIMITER_TH[23:16]	R/W	8'h80	threshold_in_db = 20 * log10 (alc_limiter_th * 2 ⁻²³)
	[8]	ALC_ATK_HOLD_RELEASE_EN	R/W	1'b0	enable control for releasing the attack hold when ALC is in atk_hold mode <ul style="list-style-type: none"> 1'b0: atk_hold is not released 1'b1: atk_hold is released
	[10:9]	ALC_ATK_HOLD_RELEASE_TH	R/W	2'h1	atk_hold release time control for ALC <ul style="list-style-type: none"> 2'b00: 5ms 2'b01: 400ms 2'b10: 1200ms 2'b11: 2400msec (@ fs=48k)
	[12:11]	ALC_ATK_HOLD_RECOV_TH	R/W	2'h1	atk_hold recover time control for ALC <ul style="list-style-type: none"> 2'b00: 5ms 2'b01: 400ms 2'b10: 1200ms 2'b11: 2400ms (@ fs=48k)
	[13]	ALC_MIN_GAIN_EN	R/W	1'b0	enable control for minimum gain control in ALC <ul style="list-style-type: none"> 1'b0: ALC minimum gain is disabled 1'b1: ALC minimum gain is disabled
	[15:14]	RSVD	N/A	2'b00	Reserved

17.6.2.25 0x27

Address	Bit	Name	Access	Reset	Description
0x27	[7:0]	ALC_MIN_GAIN	R/W	8'h0	min gain for ALC <ul style="list-style-type: none"> 8'hFF: 0db 8'hFE: -0.375db ... 8'h1: -91.875db 8'h0: -95.625db
	[8]	DA_STEREO_MODE_EN	R/W	1'b0	enable control for DAC filter <ul style="list-style-type: none"> 1'b0: ALC is mono mode 1'b1: ALC is stereo mode
	[15:9]	RSVD	R/W	7'h2F	Reserved

17.6.2.26 0x28

Address	Bit	Name	Access	Reset	Description
0x28	[7:0]	ALC_GAIN_OUT	R	8'h0	ALC output gain
	[8]	ALC_NOISE_GATE_MODE_L	R	1'b0	Status of noise_gate mode in L channel <ul style="list-style-type: none"> 1'b0: Not noise_gate mode 1'b1: noise_gate mode
	[9]	ALC_NOISE_GATE_MODE_R	R	1'b0	Status of noise_gate mode in R channel <ul style="list-style-type: none"> 1'b0: Not noise_gate mode 1'b1: noise_gate mode
	[10]	ALC_OP_MODE_L	R	1'b0	Status of ALC operation mode in L channel <ul style="list-style-type: none"> 1'b0: Recover mode 1'b1: Attack mode
	[11]	ALC_OP_MODE_R	R	1'b0	Status of ALC operation mode in R channel <ul style="list-style-type: none"> 1'b0: Recover mode

					<ul style="list-style-type: none"> 1'b1: Attack mode
	[12]	ALC_RC_MODE_L	R	1'b0	Recovery status of L channel <ul style="list-style-type: none"> 1'b0: Slow recover 1'b1: Fast recover
	[13]	ALC_RC_MODE_R	R	1'b0	Recovery status of R channel <ul style="list-style-type: none"> 1'b0: Slow recover 1'b1: Fast recover
	[14]	ALC_ZERO_DATA_MODE_L	R	1'b0	Status of zero_data mode in L channel <ul style="list-style-type: none"> 1'b0: Not zero_data mode 1'b1: zero_data mode
	[15]	ALC_ZERO_DATA_MODE_R	R	1'b0	Status of zero_data mode in R channel <ul style="list-style-type: none"> 1'b0: Not zero_data mode 1'b1: zero_data mode

17.6.2.27 0x29

Address	Bit	Name	Access	Reset	Description
0x29	[0]	ALC_BK_RMS_HOV_L	R	1'b0	Status of back-end L channel <ul style="list-style-type: none"> 1'b0: Energy of back-end <= alc_thmax 1'b1: Energy of back-end > alc_thmax
	[1]	ALC_BK_RMS_HOV_R	R	1'b0	Status of back-end R channel <ul style="list-style-type: none"> 1'b0: Energy of back-end <= alc_thmax 1'b1: Energy of back-end > alc_thmax
	[2]	ALC_BK_RMS_LOV_L	R	1'b0	Status of back-end L channel <ul style="list-style-type: none"> 1'b0: Energy of back-end <= alc_thmax - alc_min_range 1'b1: Energy of back-end > alc_thmax - alc_min_range
	[3]	ALC_BK_RMS_LOV_R	R	1'b0	Status of back-end R channel <ul style="list-style-type: none"> 1'b0: Energy of back-end <= alc_thmax - alc_min_range 1'b1: Energy of back-end > alc_thmax - alc_min_range
	[4]	ALC_FORCE_FAST_RC_MODE	R	1'b0	Status of fast recover in front end <ul style="list-style-type: none"> 1'b0: Slow recover mode 1'b1: Force fast recover mode
	[5]	ALC_THMAX_MODE_L	R	1'b0	Status of noise_gate mode in L channel <ul style="list-style-type: none"> 1'b0: Not thmax_mode 1'b1: thmax_mode
	[6]	ALC_THMAX_MODE_R	R	1'b0	Status of noise_gate mode in R channel <ul style="list-style-type: none"> 1'b0: Not thmax_mode 1'b1: thmax_mode
	[7]	ALC_THMAX2_MODE_L	R	1'b0	Status of noise_gate mode in L channel <ul style="list-style-type: none"> 1'b0: Not thmax2_mode 1'b1: thmax2_mode
	[8]	ALC_THMAX2_MODE_R	R	1'b0	Status of noise_gate mode in R channel <ul style="list-style-type: none"> 1'b0: Not thmax2_mode 1'b1: thmax2_mode
	[9]	ALC_ATK_HOLD_FLAG_L	R	1'b0	Status of atk_hold mode for L channel in ALC
	[10]	ALC_ATK_HOLD_FLAG_R	R	1'b0	Status of atk_hold mode for R channel in ALC
	[11]	ALC_ATK_HOLD_FLAG	R	1'b0	Status of atk_hold mode in ALC
	[15:12]	RSVD	N/A	4'h0	Reserved

17.6.2.28 0xF6

Address	Bit	Name	Access	Reset	Description
0xF6	[0]	DAC_L_SILENCE_DET_MONO_EN	R/W	1'b0	dac_l_silence data detection enable control <ul style="list-style-type: none"> 1'b0: Disable dac_l_silence data detection 1'b1: Enable dac_l_silence data detection

[1]	DAC_L_SILENCE_DET_MONO_AUTO_EN	R/W	1'b1	dac_l_silence data detection threshold automatically control <ul style="list-style-type: none"> 1'b0: dac_l_silence data threshold is determined by 'dac_l_silence_mono_level_sel' 1'b1: dac_l_silence data threshold is determined by 'dac_l_silence_mono_data_bit'
[3:2]	DAC_L_SILENCE_MONO_DATA_BIT	R/W	2'h0	dac_l_silence detection input data word length <ul style="list-style-type: none"> 2'b00: 16-bit, corresponding to dac_l_silence data threshold =-78db 2'b01: 20-bit, corresponding to dac_l_silence data threshold =-102db 2'b10: 24-bit, corresponding to dac_l_silence data threshold =-126db 2'b11: 24-bit, corresponding to dac_l_silence data threshold =-126db
[6:4]	DAC_L_SILENCE_MONO_LEVEL_SEL	R/W	3'h1	dac_l_silence data detection threshold register control <ul style="list-style-type: none"> 3'b000: -78db 3'b001: -84db 3'b010: -90db 3'b011: -102db 3'b100: -108db 3'b101: -114db 3'b110: -126db 3'b111: -132db
[9:7]	DAC_L_SILENCE_MONO_DEBOUNCE_SEL	R/W	3'h3	dac_l_silence data detection debounce control <ul style="list-style-type: none"> 3'b000: Debounce 80ms at sample rate 48kHz 3'b001: Debounce 160ms at sample rate 48kHz 3'b010: Debounce 320ms at sample rate 48kHz 3'b011: Debounce 640ms at sample rate 48kHz 3'b100: Debounce 1.28s at sample rate 48kHz 3'b101: Debounce 2.56s at sample rate 48kHz 3'b110: Debounce 5.12s at sample rate 48kHz 3'b111: Debounce 0.16ms at sample rate 48kHz (test mode)
[10]	DAC_L_SILENCE_DET_MONO_O	R	1'b1	dac_l_silence data status (result of silence detection) <ul style="list-style-type: none"> 1'b0: Not dac_l_silence data 1'b1: dac_l_silence is detected
[11]	DAC_L_SILENCE_DET_MONO_STATUS	R	1'b0	Ongoing status of dac_l_silence detection <ul style="list-style-type: none"> 1'b0: dac_l_silence detection is resting (clock is gating) 1'b1: dac_l_silence detection is working
[15:12]	RSVD	N/A	4'h0	Reserved

17.6.2.29 0xF7

Address	Bit	Name	Access	Reset	Description
0xF7	[0]	DAC_R_SILENCE_DET_MONO_EN	R/W	1'b0	dac_r_silence data detection enable control <ul style="list-style-type: none"> 1'b0: Disable dac_r_silence data detection 1'b1: Enable dac_r_silence data detection
	[1]	DAC_R_SILENCE_DET_MONO_AUTO_EN	R/W	1'b1	dac_r_silence data detection threshold automatically control <ul style="list-style-type: none"> 1'b0: dac_r_silence data threshold is determined by 'dac_r_silence_mono_level_sel' 1'b1: dac_r_silence data threshold is determined by 'dac_r_silence_mono_data_bit'
	[3:2]	DAC_R_SILENCE_MONO_DATA_BIT	R/W	2'h0	dac_r_silence detection input data word length <ul style="list-style-type: none"> 2'b00: 16-bit, corresponding to dac_r_silence data threshold =-78db 2'b01: 20-bit, corresponding to dac_r_silence data threshold =-102db

					<ul style="list-style-type: none"> 2'b10: 24-bit, corresponding to dac_r_silence data threshold =-126db 2'b11: 24-bit, corresponding to dac_r_silence data threshold =-126db
	[6:4]	DAC_R_SILENCE_MONO_LEVEL_SEL	R/W	3'h1	dac_r_silence data detection threshold register control <ul style="list-style-type: none"> 3'b000: -78db 3'b001: -84db 3'b010: -90db 3'b011: -102db 3'b100: -108db 3'b101: -114db 3'b110: -126db 3'b111: -132db
	[9:7]	DAC_R_SILENCE_MONO_DEBOUNCE_SEL	R/W	3'h3	dac_r_silence data detection debounce control <ul style="list-style-type: none"> 3'b000: Debounce 80ms at sample rate 48kHz 3'b001: Debounce 160ms at sample rate 48kHz 3'b010: Debounce 320ms at sample rate 48kHz 3'b011: Debounce 640ms at sample rate 48kHz 3'b100: Debounce 1.28s at sample rate 48kHz 3'b101: Debounce 2.56s at sample rate 48kHz 3'b110: Debounce 5.12s at sample rate 48kHz 3'b111: Debounce 0.16ms at sample rate 48kHz (test mode)
	[10]	DAC_R_SILENCE_DET_MONO_O	R	1'b1	dac_r_silence data status (result of silence detection) <ul style="list-style-type: none"> 1'b0: Not dac_r_silence data 1'b1: dac_r_silence is detected
	[11]	DAC_R_SILENCE_DET_MONO_STATUS	R	1'b0	Ongoing status of dac_r_silence detection <ul style="list-style-type: none"> 1'b0: dac_r_silence detection is resting (clock is gating) 1'b1: dac_r_silence detection is working
	[15:12]	RSVD	N/A	4'h0	Reserved

17.6.2.30 0xF8

Address	Bit	Name	Access	Reset	Description
0xF8	[0]	ADC_L_SILENCE_DET_MONO_EN	R/W	1'b0	adc_l_silence data detection enable control <ul style="list-style-type: none"> 1'b0: Disable adc_l_silence data detection 1'b1: Enable adc_l_silence data detection
	[1]	ADC_L_SILENCE_DET_MONO_AUTO_EN	R/W	1'b1	adc_l_silence data detection threshold automatically control <ul style="list-style-type: none"> 1'b0: adc_l_silence data threshold is register controlled 1'b1: adc_l_silence data threshold is automatically controlled
	[3:2]	ADC_L_SILENCE_MONO_DATA_BIT	R/W	2'h0	adc_l_silence detection input data word length <ul style="list-style-type: none"> 2'b00: 16-bit 2'b01: 20-bit 2'b10: 24-bit 2'b11: 24-bit
	[6:4]	ADC_L_SILENCE_MONO_LEVEL_SEL	R/W	3'h1	adc_l_silence data detection threshold register control <ul style="list-style-type: none"> 3'b000: -78db 3'b001: -84db 3'b010: -90db 3'b011: -102db 3'b100: -108db 3'b101: -114db 3'b110: -126db 3'b111: -132db
	[9:7]	ADC_L_SILENCE_MONO_DEBOUNCE_SEL	R/W	3'h3	adc_l_silence data detection debounce control <ul style="list-style-type: none"> 3'b000: Debounce 80ms at sample rate 48kHz 3'b001: Debounce 160ms at sample rate 48kHz

					<ul style="list-style-type: none"> 3'b010: Debounce 320ms at sample rate 48kHz 3'b011: Debounce 640ms at sample rate 48kHz 3'b100: Debounce 1.28s at sample rate 48kHz 3'b101: Debounce 2.56s at sample rate 48kHz 3'b110: Debounce 5.12s at sample rate 48kHz 3'b111: Debounce 0.16ms at sample rate 48kHz (test mode)
	[10]	ADC_L_SILENCE_DET_MONO_O	R	1'b1	adc_l_silence data status <ul style="list-style-type: none"> 1'b0: Not adc_l_silence data 1'b1: adc_l_silence is detected
	[11]	ADC_L_SILENCE_DET_MONO_STATUS	R	1'b0	status of adc_l_silence detection <ul style="list-style-type: none"> 1'b0: adc_l_silence detection is resting (clock is gating) 1'b1: adc_l_silence detection is working
	[15:12]	RSVD	N/A	4'h0	Reserved

17.6.2.31 0xF9

Address	Bit	Name	Access	Reset	Description
0xF9	[0]	ADC_R_SILENCE_DET_MONO_EN	R/W	1'b0	adc_r_silence data detection enable control <ul style="list-style-type: none"> 1'b0: Disable adc_r_silence data detection 1'b1: Enable adc_r_silence data detection
	[1]	ADC_R_SILENCE_DET_MONO_AUTO_EN	R/W	1'b1	adc_r_silence data detection threshold automatically control <ul style="list-style-type: none"> 1'b0: adc_r_silence data threshold is register controlled 1'b1: adc_r_silence data threshold is automatically controlled
	[3:2]	ADC_R_SILENCE_MONO_DATA_BIT	R/W	2'h0	adc_r_silence detection input data word length <ul style="list-style-type: none"> 2'b00: 16-bit 2'b01: 20-bit 2'b10: 24-bit 2'b11: 24-bit
	[6:4]	ADC_R_SILENCE_MONO_LEVEL_SEL	R/W	3'h1	adc_r_silence data detection threshold register control <ul style="list-style-type: none"> 3'b000: -78db 3'b001: -84db 3'b010: -90db 3'b011: -102db 3'b100: -108db 3'b101: -114db 3'b110: -126db 3'b111: -132db
	[9:7]	ADC_R_SILENCE_MONO_DEBOUNCE_SEL	R/W	3'h3	adc_r_silence data detection debounce control <ul style="list-style-type: none"> 3'b000: Debounce 80ms at sample rate 48kHz 3'b001: Debounce 160ms at sample rate 48kHz 3'b010: Debounce 320ms at sample rate 48kHz 3'b011: Debounce 640ms at sample rate 48kHz 3'b100: Debounce 1.28s at sample rate 48kHz 3'b101: Debounce 2.56s at sample rate 48kHz 3'b110: Debounce 5.12s at sample rate 48kHz 3'b111: Debounce 0.16ms at sample rate 48kHz (test mode)
	[10]	ADC_R_SILENCE_DET_MONO_O	R	1'b1	adc_r_silence data status <ul style="list-style-type: none"> 1'b0: Not adc_r_silence data 1'b1: adc_r_silence is detected
	[11]	ADC_R_SILENCE_DET_MONO_STATUS	R	1'b0	status of adc_r_silence detection <ul style="list-style-type: none"> 1'b0: adc_r_silence detection is resting (clock is gating) 1'b1: adc_r_silence detection is working
	[15:12]	RSVD	N/A	4'h0	Reserved

17.6.2.32 0xFA

Address	Bit	Name	Access	Reset	Description
0xFA	[7:0]	DAC_L_DA_GAIN	R/W	8'hFF	Mon DAC Lch DVOL gain control (0.375dB/step) <ul style="list-style-type: none"> 8'hAF: 0dB 8'h00: -65.625dB
	[8]	DAC_L_DAHPF_EN	R/W	1'b1	Mon DAC Lch Narrow-band 1st HPF enable control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[10:9]	DAC_L_DA_DITHER_SEL	R/W	2'b10	Mon DAC Lch dither select <ul style="list-style-type: none"> 2'b00: Disable 2'b01: LSB 2'b10: LSB+1 2'b11: LSB+2
	[12:11]	DAC_L_DA_ZDET_FUNC	R/W	2'b11	Mon DAC Lch Zero detection function mode control <ul style="list-style-type: none"> 2'b00: Immediate change 2'b01: Zero detection and immediate change 2'b10: Zero detection and increase/decrease change 2'b11: N/A
	[14:13]	DAC_L_DA_ZDET_TOUT	R/W	2'b00	Mon DAC Lch Zero detection time out mode control <ul style="list-style-type: none"> 2'b00: 1024*16 samples 2'b01: 1024*32 samples 2'b10: 1024*64 samples 2'b11: 256 samples
	[15]	DAC_L_DMIX_IN_SEL	R/W	1'b0	Mon DAC Lch upsample filter input select <ul style="list-style-type: none"> 1'b0: ALC output 1'b1: Mono DAC volume output

17.6.2.33 0xFB

Address	Bit	Name	Access	Reset	Description
0xFB	[0]	DAC_L_DA_MUTE	R/W	1'b0	Mon DAC Lch DVOL mute enable <ul style="list-style-type: none"> 1'b0: Un-mute 1'b1: Mute
	[1]	DAAD_LPBK_EN	R/W	1'b0	Digital DAC & ADC loop back control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[2]	DAC_L_DMIX_MUTE_128FS_DA	R/W	1'b0	Mon DAC Lch 128fs-domain mixer da path mute enable <ul style="list-style-type: none"> 1'b0: Un-mute 1'b1: Mute
	[3]	DAC_L_DMIX_MUTE_128FS_SIDETONE	R/W	1'b0	Mon DAC Lch 128fs-domain mixer sidetone path mute enable <ul style="list-style-type: none"> 1'b0: Un-mute 1'b1: Mute
	[15:4]	RSVD	N/A	12'h0	Reserved

17.6.2.34 0xFC

Address	Bit	Name	Access	Reset	Description
0xFC	[7:0]	DAC_R_DA_GAIN	R/W	8'hFF	Mon DAC Rch DVOL gain control (0.375dB/step) <ul style="list-style-type: none"> 8'hAF: 0dB 8'h00: -65.625dB
	[8]	DAC_R_DAHPF_EN	R/W	1'b1	Mon DAC Rch Narrow-band 1st HPF enable control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable

	[10:9]	DAC_R_DA_DITHER_SEL	R/W	2'b10	Mon DAC Rch dither select <ul style="list-style-type: none"> 2'b00: disable 2'b01: LSB 2'b10: LSB+1 2'b11: LSB+2
	[12:11]	DAC_R_DA_ZDET_FUNC	R/W	2'b11	Mon DAC Rch Zero detection function mode control <ul style="list-style-type: none"> 2'b00: Immediate change 2'b01: Zero detection and immediate change 2'b10: Zero detection and increase/decrease change 2'b11: N/A
	[14:13]	DAC_R_DA_ZDET_TOUT	R/W	2'b00	Mon DAC Rch Zero detection time out mode control <ul style="list-style-type: none"> 2'b00: 1024*16 samples 2'b01: 1024*32 samples 2'b10: 1024*64 samples 2'b11: 256 samples
	[15]	DAC_R_DMIX_IN_SEL	R/W	1'b0	Mon DAC Rch upsample filter input select <ul style="list-style-type: none"> 1'b0: ALC output 1'b1: Mono DAC volume output

17.6.2.35 0xFD

Address	Bit	Name	Access	Reset	Description
0xFD	[0]	DAC_R_DA_MUTE	R/W	1'b0	Mon DAC Rch DVOL mute enable <ul style="list-style-type: none"> 1'b0: Un-mute 1'b1: Mute
	[1]	RSVD	N/A	1'b0	Reserved
	[2]	DAC_R_DMIX_MUTE_128FS_DA	R/W	1'b0	Mon DAC Rch 128fs-domain mixer da path mute enable <ul style="list-style-type: none"> 1'b0: Un-mute 1'b1: Mute
	[3]	DAC_R_DMIX_MUTE_128FS_SIDETONE	R/W	1'b0	Mon DAC Rch 128fs-domain mixer sidetone path mute enable <ul style="list-style-type: none"> 1'b0: Un-mute 1'b1: Mute
	[15:4]	RSVD	N/A	12'h0	Reserved

17.6.3 DAC_EQ

17.6.3.1 0x2A

Address	Bit	Name	Access	Reset	Description
0x2A	[0]	DAC_L_BQ_EQ_PARAM_UPDATE	R/W	1'b1	DAC Lch EQ Write "1" to send parameter update pulse <ul style="list-style-type: none"> 1'b0: Busy (Waiting for cross) 1'b1: Stand-by
	[1]	DAC_L_BQ_EQ_CD_EN	R/W	1'b1	DAC Lch EQ cross detection control <ul style="list-style-type: none"> 1'b0: Disable (Test mode) 1'b1: Enable (Normal mode)
	[3:2]	DAC_L_BQ_EQ_DITHER_SEL	R/W	2'b00	DAC Lch EQ dither control <ul style="list-style-type: none"> 2'b00: Normal 2'b01: LSB 2'b10: {LSB+1, LSB} 2'b11: {LSB+2, LSB+1, LSB}
	[4]	RSVD	N/A	1'b0	Reserved
	[5]	DAC_L_BQ_EQ_CD_FLAG	R	1'b0	DAC Lch Biquad filter cross detect status <ul style="list-style-type: none"> 1'b0: No data zero cross

					● 1'b1: Data zero cross
	[15:6]	RSVD	N/A	10'h0	Reserved

17.6.3.2 0x2B~ 0x2C

Address	Bit	Name	Access	Reset	Description
0x2B	[15:0]	DAC_L_BIQUAD_H0_1[15:0]	R/W	16'h0000	DAC Lch EQ 1st-band coefficient h0
0x2C	[12:0]	DAC_L_BIQUAD_H0_1[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.3 0x2D ~ 0x2E

Address	Bit	Name	Access	Reset	Description
0x2D	[15:0]	DAC_L_BIQUAD_B1_1[15:0]	R/W	16'h0	DAC Lch EQ 1st-band coefficient b1
0x2E	[12:0]	DAC_L_BIQUAD_B1_1[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.4 0x2F ~ 0x30

Address	Bit	Name	Access	Reset	Description
0x2F	[15:0]	DAC_L_BIQUAD_B2_1[15:0]	R/W	16'h0	DAC Lch EQ 1st-band coefficient b2
0x30	[12:0]	DAC_L_BIQUAD_B2_1[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.5 0x31 ~ 0x32

Address	Bit	Name	Access	Reset	Description
0x31	[15:0]	DAC_L_BIQUAD_A1_1[15:0]	R/W	16'h0	DAC Lch EQ 1st-band coefficient a1
0x32	[12:0]	DAC_L_BIQUAD_A1_1[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.6 0x33 ~ 0x34

Address	Bit	Name	Access	Reset	Description
0x33	[15:0]	DAC_L_BIQUAD_A2_1[15:0]	R/W	16'h0	DAC Lch EQ 1st-band coefficient a2
0x34	[12:0]	DAC_L_BIQUAD_A2_1[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[13]	DAC_L_BIQUAD_EN_1	R/W	1'b0	DAC Lch 1st-band EQ biquad control ● 1'b0: Disable ● 1'b1: Enable
	[14]	DAC_L_BIQUAD_WCLR_1	W	1'b0	DAC Lch 1st-band Biquad filter write clear ● 1'b0: Normal ● 1'b1: Overflow Write "1" to send clear status pulse
	[15]	DAC_L_BIQUAD_STATUS_1	R	1'b0	DAC Lch 1st-band Biquad filter status ● 1'b0: Normal ● 1'b1: Overflow

17.6.3.7 0x35 ~ 0x36

Address	Bit	Name	Access	Reset	Description
0x35	[15:0]	DAC_L_BIQUAD_H0_2[15:0]	R/W	16'h0000	DAC Lch EQ 2nd-band coefficient h0 2's complement in 4.25 format, i.e. the range is from -8~7.99.
0x36	[12:0]	DAC_L_BIQUAD_H0_2[28:16]	R/W	13'h0200	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.8 0x37 ~ 0x38

Address	Bit	Name	Access	Reset	Description
0x37	[15:0]	DAC_L_BIQUAD_B1_2[15:0]	R/W	16'h0	DAC Lch EQ 2nd-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x38	[12:0]	DAC_L_BIQUAD_B1_2[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.9 0x39 ~ 0x3A

Address	Bit	Name	Access	Reset	Description
0x39	[15:0]	DAC_L_BIQUAD_B2_2[15:0]	R/W	16'h0	DAC Lch EQ 2nd-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x3A	[12:0]	DAC_L_BIQUAD_B2_2[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.10 0x3B ~ 0x3C

Address	Bit	Name	Access	Reset	Description
0x3B	[15:0]	DAC_L_BIQUAD_A1_2[15:0]	R/W	16'h0	DAC Lch EQ 2nd-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x3C	[12:0]	DAC_L_BIQUAD_A1_2[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.11 0x3D ~ 0x3E

Address	Bit	Name	Access	Reset	Description
0x3D	[15:0]	DAC_L_BIQUAD_A2_2[15:0]	R/W	16'h0	DAC Lch EQ 2nd-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x3E	[12:0]	DAC_L_BIQUAD_A2_2[28:16]	R/W	13'h0	
	[13]	DAC_L_BIQUAD_EN_2	R/W	1'b0	DAC Lch 2nd-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	DAC_L_BIQUAD_WCLR_2	W	1'b0	DAC Lch 2nd-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	DAC_L_BIQUAD_STATUS_2	R	1'b0	DAC Lch 2nd-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.3.12 0x3F ~ 0x40

Address	Bit	Name	Access	Reset	Description
0x3F	[15:0]	DAC_L_BIQUAD_H0_3[15:0]	R/W	16'h0000	DAC Lch EQ 3rd-band coefficient h0

0x40	[12:0]	DAC_L_BIQUAD_H0_3[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.13 0x41 ~ 0x42

Address	Bit	Name	Access	Reset	Description
0x41	[15:0]	DAC_L_BIQUAD_B1_3[15:0]	R/W	16'h0	DAC Lch EQ 3rd-band coefficient b1
0x42	[12:0]	DAC_L_BIQUAD_B1_3[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.14 0x43 ~ 0x44

Address	Bit	Name	Access	Reset	Description
0x43	[15:0]	DAC_L_BIQUAD_B2_3[15:0]	R/W	16'h0	DAC Lch EQ 3rd-band coefficient b2
0x44	[12:0]	DAC_L_BIQUAD_B2_3[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.15 0x45 ~ 0x46

Address	Bit	Name	Access	Reset	Description
0x45	[15:0]	DAC_L_BIQUAD_A1_3[15:0]	R/W	16'h0	DAC Lch EQ 3rd-band coefficient a1
0x46	[12:0]	DAC_L_BIQUAD_A1_3[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.16 0x47 ~ 0x48

Address	Bit	Name	Access	Reset	Description
0x47	[15:0]	DAC_L_BIQUAD_A2_3[15:0]	R/W	16'h0	DAC Lch EQ 3rd-band coefficient a2
0x48	[12:0]	DAC_L_BIQUAD_A2_3[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[13]	DAC_L_BIQUAD_EN_3	R/W	1'b0	DAC Lch 3rd-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	DAC_L_BIQUAD_WCLR_3	W	1'b0	DAC Lch 3rd-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	DAC_L_BIQUAD_STATUS_3	R	1'b0	DAC Lch 3rd-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.3.17 0x49 ~ 0x4A

Address	Bit	Name	Access	Reset	Description
0x49	[15:0]	DAC_L_BIQUAD_H0_4[15:0]	R/W	16'h0000	DAC Lch EQ 4th-band coefficient h0
0x4A	[12:0]	DAC_L_BIQUAD_H0_4[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.18 0x4B ~ 0x4C

Address	Bit	Name	Access	Reset	Description
0x4B	[15:0]	DAC_L_BIQUAD_B1_4[15:0]	R/W	16'h0	DAC Lch EQ 4th-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x4C	[12:0]	DAC_L_BIQUAD_B1_4[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.19 0x4D ~ 0x4E

Address	Bit	Name	Access	Reset	Description
0x4D	[15:0]	DAC_L_BIQUAD_B2_4[15:0]	R/W	16'h0	DAC Lch EQ 4th-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x4E	[12:0]	DAC_L_BIQUAD_B2_4[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.20 0x4F ~ 0x50

Address	Bit	Name	Access	Reset	Description
0x4F	[15:0]	DAC_L_BIQUAD_A1_4[15:0]	R/W	16'h0	DAC Lch EQ 4th-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x50	[12:0]	DAC_L_BIQUAD_A1_4[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.21 0x51 ~ 0x52

Address	Bit	Name	Access	Reset	Description
0x51	[15:0]	DAC_L_BIQUAD_A2_4[15:0]	R/W	16'h0	DAC Lch EQ 4th-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x52	[12:0]	DAC_L_BIQUAD_A2_4[28:16]	R/W	13'h0	
	[13]	DAC_L_BIQUAD_EN_4	R/W	1'b0	DAC Lch 4th-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	DAC_L_BIQUAD_WCLR_4	W	1'b0	DAC Lch 4th-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	DAC_L_BIQUAD_STATUS_4	R	1'b0	DAC Lch 4th-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.3.22 0x53 ~ 0x54

Address	Bit	Name	Access	Reset	Description
0x53	[15:0]	DAC_L_BIQUAD_H0_5[15:0]	R/W	16'h0000	DAC Lch EQ 5th-band coefficient h0 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x54	[12:0]	DAC_L_BIQUAD_H0_5[28:16]	R/W	13'h0200	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.23 0x55 ~ 0x56

Address	Bit	Name	Access	Reset	Description
0x55	[15:0]	DAC_L_BIQUAD_B1_5[15:0]	R/W	16'h0	DAC Lch EQ 5th-band coefficient b1

0x56	[12:0]	DAC_L_BIQUAD_B1_5[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.24 0x57 ~ 0x58

Address	Bit	Name	Access	Reset	Description
0x57	[15:0]	DAC_L_BIQUAD_B2_5[15:0]	R/W	16'h0	DAC Lch EQ 5th-band coefficient b2
0x58	[12:0]	DAC_L_BIQUAD_B2_5[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.25 0x59 ~ 0x5A

Address	Bit	Name	Access	Reset	Description
0x59	[15:0]	DAC_L_BIQUAD_A1_5[15:0]	R/W	16'h0	DAC Lch EQ 5th-band coefficient a1
0x5A	[12:0]	DAC_L_BIQUAD_A1_5[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.26 0x5B ~ 0x5C

Address	Bit	Name	Access	Reset	Description
0x5B	[15:0]	DAC_L_BIQUAD_A2_5[15:0]	R/W	16'h0	DAC Lch EQ 5th-band coefficient a2
0x5C	[12:0]	DAC_L_BIQUAD_A2_5[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[13]	DAC_L_BIQUAD_EN_5	R/W	1'b0	DAC Lch 5th-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	DAC_L_BIQUAD_WCLR_5	W	1'b0	DAC Lch 5th-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	DAC_L_BIQUAD_STATUS_5	R	1'b0	DAC Lch 5th-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.3.27 0x5D

Address	Bit	Name	Access	Reset	Description
0x5D	[0]	DAC_R_BQ_EQ_PARAM_UPDATE	R/W	1'b1	DAC Rch EQ <ul style="list-style-type: none"> 1'b0: Busy (Waiting for cross) 1'b1: Stand-by Write "1" to send parameter update pulse
	[1]	DAC_R_BQ_EQ_CD_EN	R/W	1'b1	DAC Rch EQ cross detection control <ul style="list-style-type: none"> 1'b0: Disable (Test mode) 1'b1: Enable (Normal mode)
	[3:2]	DAC_R_BQ_EQ_DITHER_SEL	R/W	2'b00	DAC Rch EQ dither control <ul style="list-style-type: none"> 2'b00: Normal 2'b01: LSB 2'b10: {LSB+1, LSB} 2'b11: {LSB+2, LSB+1, LSB}
	[4]	RSVD	N/A	1'b0	Reserved
	[5]	DAC_R_BQ_EQ_CD_FLAG	R		DAC Rch Biquad filter cross detect status

					<ul style="list-style-type: none"> 1'b0: No data zero cross 1'b1: Data zero cross
	[15:6]	RSVD	N/A	10'h0	Reserved

17.6.3.28 0x5E ~ 0x5F

Address	Bit	Name	Access	Reset	Description
0x5E	[15:0]	DAC_R_BIQUAD_H0_1[15:0]	R/W	16'h0000	DAC Rch EQ 1st-band coefficient h0
0x5F	[12:0]	DAC_R_BIQUAD_H0_1[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.29 0x60 ~ 0x61

Address	Bit	Name	Access	Reset	Description
0x60	[15:0]	DAC_R_BIQUAD_B1_1[15:0]	R/W	16'h0	DAC Rch EQ 1st-band coefficient b1
0x61	[12:0]	DAC_R_BIQUAD_B1_1[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.30 0x62 ~ 0x63

Address	Bit	Name	Access	Reset	Description
0x62	[15:0]	DAC_R_BIQUAD_B2_1[15:0]	R/W	16'h0	DAC Rch EQ 1st-band coefficient b2
0x63	[12:0]	DAC_R_BIQUAD_B2_1[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.31 0x64 ~ 0x65

Address	Bit	Name	Access	Reset	Description
0x64	[15:0]	DAC_R_BIQUAD_A1_1[15:0]	R/W	16'h0	DAC Rch EQ 1st-band coefficient a1
0x65	[12:0]	DAC_R_BIQUAD_A1_1[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.32 0x66 ~ 0x67

Address	Bit	Name	Access	Reset	Description
0x66	[15:0]	DAC_R_BIQUAD_A2_1[15:0]	R/W	16'h0	DAC Rch EQ 1st-band coefficient a2
0x67	[12:0]	DAC_R_BIQUAD_A2_1[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that. the range is from -8~7.99.
	[13]	DAC_R_BIQUAD_EN_1	R/W	1'b0	DAC Rch 1st-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	DAC_R_BIQUAD_WCLR_1	W	1'b0	DAC Rch 1st-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	DAC_R_BIQUAD_STATUS_1	R	1'b0	DAC Rch 1st-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.3.33 0x68 ~ 0x69

Address	Bit	Name	Access	Reset	Description
0x68	[15:0]	DAC_R_BIQUAD_H0_2[15:0]	R/W	16'h0000	DAC Rch EQ 2nd-band coefficient h0 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x69	[12:0]	DAC_R_BIQUAD_H0_2[28:16]	R/W	13'h0200	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.34 0x6A ~ 0x6B

Address	Bit	Name	Access	Reset	Description
0x6A	[15:0]	DAC_R_BIQUAD_B1_2[15:0]	R/W	16'h0	DAC Rch EQ 2nd-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x6B	[12:0]	DAC_R_BIQUAD_B1_2[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.35 0x6C ~ 0x6D

Address	Bit	Name	Access	Reset	Description
0x6C	[15:0]	DAC_R_BIQUAD_B2_2[15:0]	R/W	16'h0	DAC Rch EQ 2nd-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x6D	[12:0]	DAC_R_BIQUAD_B2_2[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.36 0x6E ~ 0x6F

Address	Bit	Name	Access	Reset	Description
0x6E	[15:0]	DAC_R_BIQUAD_A1_2[15:0]	R/W	16'h0	DAC Rch EQ 2nd-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x6F	[12:0]	DAC_R_BIQUAD_A1_2[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.37 0x70 ~ 0x71

Address	Bit	Name	Access	Reset	Description
0x70	[15:0]	DAC_R_BIQUAD_A2_2[15:0]	R/W	16'h0	DAC Rch EQ 2nd-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x71	[12:0]	DAC_R_BIQUAD_A2_2[28:16]	R/W	13'h0	
	[13]	DAC_R_BIQUAD_EN_2	R/W	1'b0	DAC Rch 2nd-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	DAC_R_BIQUAD_WCLR_2	W	1'b0	DAC Rch 2nd-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	DAC_R_BIQUAD_STATUS_2	R	1'b0	DAC Rch 2nd-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.3.38 0x72 ~ 0x73

Address	Bit	Name	Access	Reset	Description
0x72	[15:0]	DAC_R_BIQUAD_H0_3[15:0]	R/W	16'h0000	DAC Rch EQ 3rd-band coefficient h0

0x73	[12:0]	DAC_R_BIQUAD_H0_3[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.39 0x74 ~ 0x75

Address	Bit	Name	Access	Reset	Description
0x74	[15:0]	DAC_R_BIQUAD_B1_3[15:0]	R/W	16'h0	DAC Rch EQ 3rd-band coefficient b1
0x75	[12:0]	DAC_R_BIQUAD_B1_3[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.40 0x76 ~ 0x77

Address	Bit	Name	Access	Reset	Description
0x76	[15:0]	DAC_R_BIQUAD_B2_3[15:0]	R/W	16'h0	DAC Rch EQ 3rd-band coefficient b2
0x77	[12:0]	DAC_R_BIQUAD_B2_3[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.41 0x78 ~ 0x79

Address	Bit	Name	Access	Reset	Description
0x78	[15:0]	DAC_R_BIQUAD_A1_3[15:0]	R/W	16'h0	DAC Rch EQ 3rd-band coefficient a1
0x79	[12:0]	DAC_R_BIQUAD_A1_3[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.42 0x7A ~ 0x7B

Address	Bit	Name	Access	Reset	Description
0x7A	[15:0]	DAC_R_BIQUAD_A2_3[15:0]	R/W	16'h0	DAC Rch EQ 3rd-band coefficient a2
0x7B	[12:0]	DAC_R_BIQUAD_A2_3[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[13]	DAC_R_BIQUAD_EN_3	R/W	1'b0	DAC Rch 3rd-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	DAC_R_BIQUAD_WCLR_3	W	1'b0	DAC Rch 3rd-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	DAC_R_BIQUAD_STATUS_3	R	1'b0	DAC Rch 3rd-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.3.43 0x7C ~ 0x7D

Address	Bit	Name	Access	Reset	Description
0x7C	[15:0]	DAC_R_BIQUAD_H0_4[15:0]	R/W	16'h0000	DAC Rch EQ 4th-band coefficient h0
0x7D	[12:0]	DAC_R_BIQUAD_H0_4[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.44 0x7E ~ 0x7F

Address	Bit	Name	Access	Reset	Description
0x7E	[15:0]	DAC_R_BIQUAD_B1_4[15:0]	R/W	16'h0	DAC Rch EQ 4th-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x7F	[12:0]	DAC_R_BIQUAD_B1_4[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.45 0x80 ~ 0x81

Address	Bit	Name	Access	Reset	Description
0x80	[15:0]	DAC_R_BIQUAD_B2_4[15:0]	R/W	16'h0	DAC Rch EQ 4th-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x81	[12:0]	DAC_R_BIQUAD_B2_4[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.46 0x82 ~ 0x83

Address	Bit	Name	Access	Reset	Description
0x82	[15:0]	DAC_R_BIQUAD_A1_4[15:0]	R/W	16'h0	DAC Rch EQ 4th-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x83	[12:0]	DAC_R_BIQUAD_A1_4[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.47 0x84 ~ 0x85

Address	Bit	Name	Access	Reset	Description
0x84	[15:0]	DAC_R_BIQUAD_A2_4[15:0]	R/W	16'h0	DAC Rch EQ 4th-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x85	[12:0]	DAC_R_BIQUAD_A2_4[28:16]	R/W	13'h0	
	[13]	DAC_R_BIQUAD_EN_4	R/W	1'b0	DAC Rch 4th-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	DAC_R_BIQUAD_WCLR_4	W	1'b0	DAC Rch 4th-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	DAC_R_BIQUAD_STATUS_4	R	1'b0	DAC Rch 4th-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.3.48 0x86 ~ 0x87

Address	Bit	Name	Access	Reset	Description
0x86	[15:0]	DAC_R_BIQUAD_H0_5[15:0]	R/W	16'h0000	DAC Rch EQ 5th-band coefficient h0 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x87	[12:0]	DAC_R_BIQUAD_H0_5[28:16]	R/W	13'h0200	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.49 0x88 ~ 0x89

Address	Bit	Name	Access	Reset	Description
0x88	[15:0]	DAC_R_BIQUAD_B1_5[15:0]	R/W	16'h0	DAC Rch EQ 5th-band coefficient b1

0x89	[12:0]	DAC_R_BIQUAD_B1_5[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.50 0x8A ~ 0x8B

Address	Bit	Name	Access	Reset	Description
0x8A	[15:0]	DAC_R_BIQUAD_B2_5[15:0]	R/W	16'h0	DAC Rch EQ 5th-band coefficient b2
0x8B	[12:0]	DAC_R_BIQUAD_B2_5[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.51 0x8C ~ 0x8D

Address	Bit	Name	Access	Reset	Description
0x8C	[15:0]	DAC_R_BIQUAD_A1_5[15:0]	R/W	16'h0	DAC Rch EQ 5th-band coefficient a1
0x8D	[12:0]	DAC_R_BIQUAD_A1_5[28:16]	R/W	13'h0	2's complement in 4.25 format, i.e. the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.3.52 0x8E ~ 0x8F

Address	Bit	Name	Access	Reset	Description
0x8E	[15:0]	DAC_R_BIQUAD_A2_5[15:0]	R/W	16'h0	DAC Rch EQ 5th-band coefficient a2
0x8F	[12:0]	DAC_R_BIQUAD_A2_5[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[13]	DAC_R_BIQUAD_EN_5	R/W	1'b0	DAC Rch 5th-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	DAC_R_BIQUAD_WCLR_5	W	1'b0	DAC Rch 5th-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	DAC_R_BIQUAD_STATUS_5	R	1'b0	DAC Rch 5th-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.4 ADC_EQ

17.6.4.1 0x90

Address	Bit	Name	Access	Reset	Description
0x90	[0]	ADC_L_BQ_EQ_PARAM_UPDATE	R/W	1'b1	ADC Lch EQ <ul style="list-style-type: none"> 1'b0: Busy (Waiting for cross) 1'b1: Stand-by Write "1" to send parameter update pulse
	[1]	ADC_L_BQ_EQ_CD_EN	R/W	1'b1	ADC Lch EQ cross detection control <ul style="list-style-type: none"> 1'b0: Disable (Test mode) 1'b1: Enable (Normal mode)
	[3:2]	ADC_L_BQ_EQ_DITHER_SEL	R/W	2'b00	ADC Lch EQ dither control <ul style="list-style-type: none"> 2'b00: Normal 2'b01: LSB 2'b10: {LSB+1, LSB}

					<ul style="list-style-type: none"> 2'b11: {LSB+2, LSB+1, LSB}
	[4]	RSVD	N/A	1'b0	Reserved
	[5]	ADC_L_BQ_EQ_CD_FLAG	R	1'b0	ADC Lch Biquad filter cross detect status <ul style="list-style-type: none"> 1'b0: No data zero cross 1'b1: Data zero cross
	[15:6]	RSVD	N/A	10'h0	Reserved

17.6.4.2 0x91 ~ 0x92

Address	Bit	Name	Access	Reset	Description
0x91	[15:0]	ADC_L_BIQUAD_HO_1[15:0]	R/W	16'h0000	ADC Lch EQ 1st-band coefficient h0 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x92	[12:0]	ADC_L_BIQUAD_HO_1[28:16]	R/W	13'h0200	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.3 0x93 ~ 0x94

Address	Bit	Name	Access	Reset	Description
0x93	[15:0]	ADC_L_BIQUAD_B1_1[15:0]	R/W	16'h0	ADC Lch EQ 1st-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x94	[12:0]	ADC_L_BIQUAD_B1_1[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.4 0x95 ~ 0x96

Address	Bit	Name	Access	Reset	Description
0x95	[15:0]	ADC_L_BIQUAD_B2_1[15:0]	R/W	16'h0	ADC Lch EQ 1st-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x96	[12:0]	ADC_L_BIQUAD_B2_1[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.5 0x97 ~ 0x98

Address	Bit	Name	Access	Reset	Description
0x97	[15:0]	ADC_L_BIQUAD_A1_1[15:0]	R/W	16'h0	ADC Lch EQ 1st-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x98	[12:0]	ADC_L_BIQUAD_A1_1[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.6 0x99 ~ 0x9A

Address	Bit	Name	Access	Reset	Description
0x99	[15:0]	ADC_L_BIQUAD_A2_1[15:0]	R/W	16'h0	ADC Lch EQ 1st-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0x9A	[12:0]	ADC_L_BIQUAD_A2_1[28:16]	R/W	13'h0	
	[13]	ADC_L_BIQUAD_EN_1	R/W	1'b0	ADC Lch 1st-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	ADC_L_BIQUAD_WCLR_1	W	1'b0	ADC Lch 1st-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	ADC_L_BIQUAD_STATUS_1	R	1'b0	ADC Lch 1st-band Biquad filter status

					<ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow
--	--	--	--	--	--

17.6.4.7 0x9B ~ 0x9C

Address	Bit	Name	Access	Reset	Description
0x9B	[15:0]	ADC_L_BIQUAD_H0_2[15:0]	R/W	16'h0000	ADC Lch EQ 2nd-band coefficient h0
0x9C	[12:0]	ADC_L_BIQUAD_H0_2[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.8 0x9D ~ 0x9E

Address	Bit	Name	Access	Reset	Description
0x9D	[15:0]	ADC_L_BIQUAD_B1_2[15:0]	R/W	16'h0	ADC Lch EQ 2nd-band coefficient b1
0x9E	[12:0]	ADC_L_BIQUAD_B1_2[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.9 0x9F ~ 0xA0

Address	Bit	Name	Access	Reset	Description
0x9F	[15:0]	ADC_L_BIQUAD_B2_2[15:0]	R/W	16'h0	ADC Lch EQ 2nd-band coefficient b2
0xA0	[12:0]	ADC_L_BIQUAD_B2_2[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.10 0xA1 ~ 0xA2

Address	Bit	Name	Access	Reset	Description
0xA1	[15:0]	ADC_L_BIQUAD_A1_2[15:0]	R/W	16'h0	ADC Lch EQ 2nd-band coefficient a1
0xA2	[12:0]	ADC_L_BIQUAD_A1_2[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.11 0xA3 ~ 0xA4

Address	Bit	Name	Access	Reset	Description
0xA3	[15:0]	ADC_L_BIQUAD_A2_2[15:0]	R/W	16'h0	ADC Lch EQ 2nd-band coefficient a2
0xA4	[12:0]	ADC_L_BIQUAD_A2_2[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[13]	ADC_L_BIQUAD_EN_2	R/W	1'b0	ADC Lch 2nd-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	ADC_L_BIQUAD_WCLR_2	W	1'b0	ADC Lch 2nd-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	ADC_L_BIQUAD_STATUS_2	R	1'b0	ADC Lch 2nd-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.4.12 0xA5 ~ 0xA6

Address	Bit	Name	Access	Reset	Description
0xA5	[15:0]	ADC_L_BIQUAD_H0_3[15:0]	R/W	16'h0000	ADC Lch EQ 3rd-band coefficient h0 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xA6	[12:0]	ADC_L_BIQUAD_H0_3[28:16]	R/W	13'h0200	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.13 0xA7 ~ 0xA8

Address	Bit	Name	Access	Reset	Description
0xA7	[15:0]	ADC_L_BIQUAD_B1_3[15:0]	R/W	16'h0	ADC Lch EQ 3rd-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xA8	[12:0]	ADC_L_BIQUAD_B1_3[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.14 0xA9 ~ 0xAA

Address	Bit	Name	Access	Reset	Description
0xA9	[15:0]	ADC_L_BIQUAD_B2_3[15:0]	R/W	16'h0	ADC Lch EQ 3rd-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xAA	[12:0]	ADC_L_BIQUAD_B2_3[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.15 0xAB ~ 0xAC

Address	Bit	Name	Access	Reset	Description
0xAB	[15:0]	ADC_L_BIQUAD_A1_3[15:0]	R/W	16'h0	ADC Lch EQ 3rd-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xAC	[12:0]	ADC_L_BIQUAD_A1_3[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.16 0xAD ~ 0xAE

Address	Bit	Name	Access	Reset	Description
0xAD	[15:0]	ADC_L_BIQUAD_A2_3[15:0]	R/W	16'h0	ADC Lch EQ 3rd-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xAE	[12:0]	ADC_L_BIQUAD_A2_3[28:16]	R/W	13'h0	
	[13]	ADC_L_BIQUAD_EN_3	R/W	1'b0	ADC Lch 3rd-band EQ Biquad control ● 1'b0: Disable ● 1'b1: Enable
	[14]	ADC_L_BIQUAD_WCLR_3	W	1'b0	ADC Lch 3rd-band Biquad filter write clear ● 1'b0: Normal ● 1'b1: Overflow Write "1" to send clear status pulse
	[15]	ADC_L_BIQUAD_STATUS_3	R	1'b0	ADC Lch 3rd-band Biquad filter status ● 1'b0: Normal ● 1'b1: Overflow

17.6.4.17 0xAF ~ 0xB0

Address	Bit	Name	Access	Reset	Description
0xAF	[15:0]	ADC_L_BIQUAD_H0_4[15:0]	R/W	16'h0000	ADC Lch EQ 4th-band coefficient h0

0xB0	[12:0]	ADC_L_BIQUAD_H0_4[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.18 0xB1 ~ 0xB2

Address	Bit	Name	Access	Reset	Description
0xB1	[15:0]	ADC_L_BIQUAD_B1_4[15:0]	R/W	16'h0	ADC Lch EQ 4th-band coefficient b1
0xB2	[12:0]	ADC_L_BIQUAD_B1_4[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.19 0xB3 ~ 0xB4

Address	Bit	Name	Access	Reset	Description
0xB3	[15:0]	ADC_L_BIQUAD_B2_4[15:0]	R/W	16'h0	ADC Lch EQ 4th-band coefficient b2
0xB4	[12:0]	ADC_L_BIQUAD_B2_4[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.20 0xB5 ~ 0xB6

Address	Bit	Name	Access	Reset	Description
0xB5	[15:0]	ADC_L_BIQUAD_A1_4[15:0]	R/W	16'h0	ADC Lch EQ 4th-band coefficient a1
0xB6	[12:0]	ADC_L_BIQUAD_A1_4[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.21 0xB7 ~ 0xB8

Address	Bit	Name	Access	Reset	Description
0xB7	[15:0]	ADC_L_BIQUAD_A2_4[15:0]	R/W	16'h0	ADC Lch EQ 4th-band coefficient a2
0xB8	[12:0]	ADC_L_BIQUAD_A2_4[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[13]	ADC_L_BIQUAD_EN_4	R/W	1'b0	ADC Lch 4th-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	ADC_L_BIQUAD_WCLR_4	W	1'b0	ADC Lch 4th-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	ADC_L_BIQUAD_STATUS_4	R	1'b0	ADC Lch 4th-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.4.22 0xB9 ~ 0xBA

Address	Bit	Name	Access	Reset	Description
0xB9	[15:0]	ADC_L_BIQUAD_H0_5[15:0]	R/W	16'h0000	ADC Lch EQ 5th-band coefficient h0
0xBA	[12:0]	ADC_L_BIQUAD_H0_5[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.23 0xBB ~ 0xBC

Address	Bit	Name	Access	Reset	Description
0xBB	[15:0]	ADC_L_BIQUAD_B1_5[15:0]	R/W	16'h0	ADC Lch EQ 5th-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xBC	[12:0]	ADC_L_BIQUAD_B1_5[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.24 0xBD ~ 0xBE

Address	Bit	Name	Access	Reset	Description
0xBD	[15:0]	ADC_L_BIQUAD_B2_5[15:0]	R/W	16'h0	ADC Lch EQ 5th-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xBE	[12:0]	ADC_L_BIQUAD_B2_5[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.25 0xBF ~ 0xC0

Address	Bit	Name	Access	Reset	Description
0xBF	[15:0]	ADC_L_BIQUAD_A1_5[15:0]	R/W	16'h0	ADC Lch EQ 5th-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xC0	[12:0]	ADC_L_BIQUAD_A1_5[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.26 0xC1 ~ 0xC2

Address	Bit	Name	Access	Reset	Description
0xC1	[15:0]	ADC_L_BIQUAD_A2_5[15:0]	R/W	16'h0	ADC Lch EQ 5th-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xC2	[12:0]	ADC_L_BIQUAD_A2_5[28:16]	R/W	13'h0	
	[13]	ADC_L_BIQUAD_EN_5	R/W	1'b0	ADC Lch 5th-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	ADC_L_BIQUAD_WCLR_5	W	1'b0	ADC Lch 5th-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Writing 1 to send clear status pulse
	[15]	ADC_L_BIQUAD_STATUS_5	R	1'b0	ADC Lch 5th-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.4.27 0xC3

Address	Bit	Name	Access	Reset	Description
0xC3	[0]	ADC_R_BQ_EQ_PARAM_UPDATE	R/W	1'b1	ADC Rch EQ <ul style="list-style-type: none"> 1'b0: Busy (Waiting for cross) 1'b1: Stand-by Write "1" to send parameter update pulse
	[1]	ADC_R_BQ_EQ_CD_EN	R/W	1'b1	ADC Rch EQ cross detection control <ul style="list-style-type: none"> 1'b0: Disable (Test mode) 1'b1: Enable (Normal mode)
	[3:2]	ADC_R_BQ_EQ_DITHER_SEL	R/W	2'b00	ADC Rch EQ dither control <ul style="list-style-type: none"> 2'b00: Normal

					<ul style="list-style-type: none"> 2'b01: LSB 2'b10: {LSB+1, LSB} 2'b11: {LSB+2, LSB+1, LSB}
	[4]	RSVD	N/A	1'b0	Reserved
	[5]	ADC_R_BQ_EQ_CD_FLAG	R	1'b0	ADC Rch Biquad filter cross detect status <ul style="list-style-type: none"> 1'b0: no data zero cross 1'b1: data zero cross
	[15:6]	RSVD	N/A	10'h0	Reserved

17.6.4.28 0xC4 ~ 0xC5

Address	Bit	Name	Access	Reset	Description
0xC4	[15:0]	ADC_R_BIQUAD_H0_1[15:0]	R/W	16'h0000	ADC Rch EQ 1st-band coefficient h0 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xC5	[12:0]	ADC_R_BIQUAD_H0_1[28:16]	R/W	13'h0200	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.29 0xC6 ~ 0xC7

Address	Bit	Name	Access	Reset	Description
0xC6	[15:0]	ADC_R_BIQUAD_B1_1[15:0]	R/W	16'h0	ADC Rch EQ 1st-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xC7	[12:0]	ADC_R_BIQUAD_B1_1[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.30 0xC8 ~ 0xC9

Address	Bit	Name	Access	Reset	Description
0xC8	[15:0]	ADC_R_BIQUAD_B2_1[15:0]	R/W	16'h0	ADC Rch EQ 1st-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xC9	[12:0]	ADC_R_BIQUAD_B2_1[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.31 0xCA ~ 0xCB

Address	Bit	Name	Access	Reset	Description
0xCA	[15:0]	ADC_R_BIQUAD_A1_1[15:0]	R/W	16'h0	ADC Rch EQ 1st-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xCB	[12:0]	ADC_R_BIQUAD_A1_1[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.32 0xCC ~ 0xCD

Address	Bit	Name	Access	Reset	Description
0xCC	[15:0]	ADC_R_BIQUAD_A2_1[15:0]	R/W	16'h0	ADC Rch EQ 1st-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xCD	[12:0]	ADC_R_BIQUAD_A2_1[28:16]	R/W	13'h0	
	[13]	ADC_R_BIQUAD_EN_1	R/W	1'b0	ADC Rch 1st-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	ADC_R_BIQUAD_WCLR_1	W	1'b0	ADC Rch 1st-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

					Write "1" to send clear status pulse
	[15]	ADC_R_BIQUAD_STATUS_1	R	1'b0	ADC Rch 1st-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.4.33 0xCE ~ 0xCF

Address	Bit	Name	Access	Reset	Description
0xCE	[15:0]	ADC_R_BIQUAD_H0_2[15:0]	R/W	16'h0000	ADC Rch EQ 2nd-band coefficient h0
0xCF	[12:0]	ADC_R_BIQUAD_H0_2[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.34 0xD0 ~ 0xD1

Address	Bit	Name	Access	Reset	Description
0xD0	[15:0]	ADC_R_BIQUAD_B1_2[15:0]	R/W	16'h0	ADC Rch EQ 2nd-band coefficient b1
0xD1	[12:0]	ADC_R_BIQUAD_B1_2[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.35 0xD2 ~ 0xD3

Address	Bit	Name	Access	Reset	Description
0xD2	[15:0]	ADC_R_BIQUAD_B2_2[15:0]	R/W	16'h0	ADC Rch EQ 2nd-band coefficient b2
0xD3	[12:0]	ADC_R_BIQUAD_B2_2[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.36 0xD4 ~ 0xD5

Address	Bit	Name	Access	Reset	Description
0xD4	[15:0]	ADC_R_BIQUAD_A1_2[15:0]	R/W	16'h0	ADC Rch EQ 2nd-band coefficient a1
0xD5	[12:0]	ADC_R_BIQUAD_A1_2[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.37 0xD6 ~ 0xD7

Address	Bit	Name	Access	Reset	Description
0xD6	[15:0]	ADC_R_BIQUAD_A2_2[15:0]	R/W	16'h0	ADC Rch EQ 2nd-band coefficient a2
0xD7	[12:0]	ADC_R_BIQUAD_A2_2[28:16]	R/W	13'h0	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[13]	ADC_R_BIQUAD_EN_2	R/W	1'b0	ADC Rch 2nd-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	ADC_R_BIQUAD_WCLR_2	W	1'b0	ADC Rch 2nd-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	ADC_R_BIQUAD_STATUS_2	R	1'b0	ADC Rch 2nd-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.4.38 0xD8 ~ 0xD9

Address	Bit	Name	Access	Reset	Description
0xD8	[15:0]	ADC_R_BIQUAD_H0_3[15:0]	R/W	16'h0000	ADC Rch EQ 3rd-band coefficient h0 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xD9	[12:0]	ADC_R_BIQUAD_H0_3[28:16]	R/W	13'h0200	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.39 0xDA ~ 0xDB

Address	Bit	Name	Access	Reset	Description
0xDA	[15:0]	ADC_R_BIQUAD_B1_3[15:0]	R/W	16'h0	ADC Rch EQ 3rd-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xDB	[12:0]	ADC_R_BIQUAD_B1_3[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.40 0xDC ~ 0xDD

Address	Bit	Name	Access	Reset	Description
0xDC	[15:0]	ADC_R_BIQUAD_B2_3[15:0]	R/W	16'h0	ADC Rch EQ 3rd-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xDD	[12:0]	ADC_R_BIQUAD_B2_3[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.41 0xDE ~ 0xDF

Address	Bit	Name	Access	Reset	Description
0xDE	[15:0]	ADC_R_BIQUAD_A1_3[15:0]	R/W	16'h0	ADC Rch EQ 3rd-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xDF	[12:0]	ADC_R_BIQUAD_A1_3[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.42 0xE0 ~ 0xE1

Address	Bit	Name	Access	Reset	Description
0xE0	[15:0]	ADC_R_BIQUAD_A2_3[15:0]	R/W	16'h0	ADC Rch EQ 3rd-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xE1	[12:0]	ADC_R_BIQUAD_A2_3[28:16]	R/W	13'h0	
	[13]	ADC_R_BIQUAD_EN_3	R/W	1'b0	ADC Rch 3rd-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	ADC_R_BIQUAD_WCLR_3	W	1'b0	ADC Rch 3rd-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	ADC_R_BIQUAD_STATUS_3	R	1'b0	ADC Rch 3rd-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

17.6.4.43 0xE2 ~ 0xE3

Address	Bit	Name	Access	Reset	Description
0xE2	[15:0]	ADC_R_BIQUAD_H0_4[15:0]	R/W	16'h0000	ADC Rch EQ 4th-band coefficient h0

0xE3	[12:0]	ADC_R_BIQUAD_H0_4[28:16]	R/W	13'h0200	2's complement in 4.25 format, which means that the range is from -8~7.99.
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.44 0xE4 ~ 0xE5

Address	Bit	Name	Access	Reset	Description
0xE4	[15:0]	ADC_R_BIQUAD_B1_4[15:0]	R/W	16'h0	ADC Rch EQ 4th-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xE5	[12:0]	ADC_R_BIQUAD_B1_4[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	

17.6.4.45 0xE6 ~ 0xE7

Address	Bit	Name	Access	Reset	Description
0xE6	[15:0]	ADC_R_BIQUAD_B2_4[15:0]	R/W	16'h0	ADC Rch EQ 4th-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xE7	[12:0]	ADC_R_BIQUAD_B2_4[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	

17.6.4.46 0xE8 ~ 0xE9

Address	Bit	Name	Access	Reset	Description
0xE8	[15:0]	ADC_R_BIQUAD_A1_4[15:0]	R/W	16'h0	ADC Rch EQ 4th-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xE9	[12:0]	ADC_R_BIQUAD_A1_4[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	

17.6.4.47 0xEA ~ 0xEB

Address	Bit	Name	Access	Reset	Description
0xEA	[15:0]	ADC_R_BIQUAD_A2_4[15:0]	R/W	16'h0	ADC Rch EQ 4th-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xEB	[12:0]	ADC_R_BIQUAD_A2_4[28:16]	R/W	13'h0	
	[13]	ADC_R_BIQUAD_EN_4	R/W	1'b0	ADC Rch 4th-band EQ Biquad control ● 1'b0: Disable ● 1'b1: Enable
	[14]	ADC_R_BIQUAD_WCLR_4	W	1'b0	ADC Rch 4th-band Biquad filter write clear ● 1'b0: Normal ● 1'b1: Overflow Write "1" to send clear status pulse
	[15]	ADC_R_BIQUAD_STATUS_4	R	1'b0	ADC Rch 4th-band Biquad filter status ● 1'b0: Normal ● 1'b1: Overflow

17.6.4.48 0xEC ~ 0xED

Address	Bit	Name	Access	Reset	Description
0xEC	[15:0]	ADC_R_BIQUAD_H0_5[15:0]	R/W	16'h0000	ADC Rch EQ 5th-band coefficient h0 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xED	[12:0]	ADC_R_BIQUAD_H0_5[28:16]	R/W	13'h0200	
	[15:13]	RSVD	N/A	3'b000	

17.6.4.49 0xEE ~ 0xEF

Address	Bit	Name	Access	Reset	Description
0xEE	[15:0]	ADC_R_BIQUAD_B1_5[15:0]	R/W	16'h0	ADC Rch EQ 5th-band coefficient b1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xEF	[12:0]	ADC_R_BIQUAD_B1_5[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.50 0xF0 ~ 0xF1

Address	Bit	Name	Access	Reset	Description
0xF0	[15:0]	ADC_R_BIQUAD_B2_5[15:0]	R/W	16'h0	ADC Rch EQ 5th-band coefficient b2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xF1	[12:0]	ADC_R_BIQUAD_B2_5[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.51 0xF2 ~ 0xF3

Address	Bit	Name	Access	Reset	Description
0xF2	[15:0]	ADC_R_BIQUAD_A1_5[15:0]	R/W	16'h0	ADC Rch EQ 5th-band coefficient a1 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xF3	[12:0]	ADC_R_BIQUAD_A1_5[28:16]	R/W	13'h0	
	[15:13]	RSVD	N/A	3'b000	Reserved

17.6.4.52 0xF4 ~ 0xF5

Address	Bit	Name	Access	Reset	Description
0xF4	[15:0]	ADC_R_BIQUAD_A2_5[15:0]	R/W	16'h0	ADC Rch EQ 5th-band coefficient a2 2's complement in 4.25 format, which means that the range is from -8~7.99.
0xF5	[12:0]	ADC_R_BIQUAD_A2_5[28:16]	R/W	13'h0	
	[13]	ADC_R_BIQUAD_EN_5	R/W	1'b0	ADC Rch 5th-band EQ Biquad control <ul style="list-style-type: none"> 1'b0: Disable 1'b1: Enable
	[14]	ADC_R_BIQUAD_WCLR_5	W	1'b0	ADC Rch 5th-band Biquad filter write clear <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow Write "1" to send clear status pulse
	[15]	ADC_R_BIQUAD_STATUS_5	R	1'b0	ADC Rch 5th-band Biquad filter status <ul style="list-style-type: none"> 1'b0: Normal 1'b1: Overflow

18 Audio Codec Controller (ACC)

18.1 Introduction

Ameba-D audio codec control (ACC) is the bridge between host audio buffers and audio codec module.

It transfers audio data to or from audio codec module via SPORT sequentially according to user settings, such as sample rate, sampling resolution, channel number, etc.

On the other hand, Ameba-D drives the audio codec module to work well by means of ACC, which is implemented via SI bus.

18.2 Features

- Mandatory or optional sample rate which audio codec module declares to support
- 8-bit/16-bit/24-bit sampling resolution
- Mono/stereo audio
- I²S/left-justified/PCM data formats
- GDMA interface for data flow
- Data loopback between SDI and SDO

18.3 Architecture

The ACC is used for audio codec input/output control. The ACC and AC architecture of Ameba-D is shown in Fig 18-1.

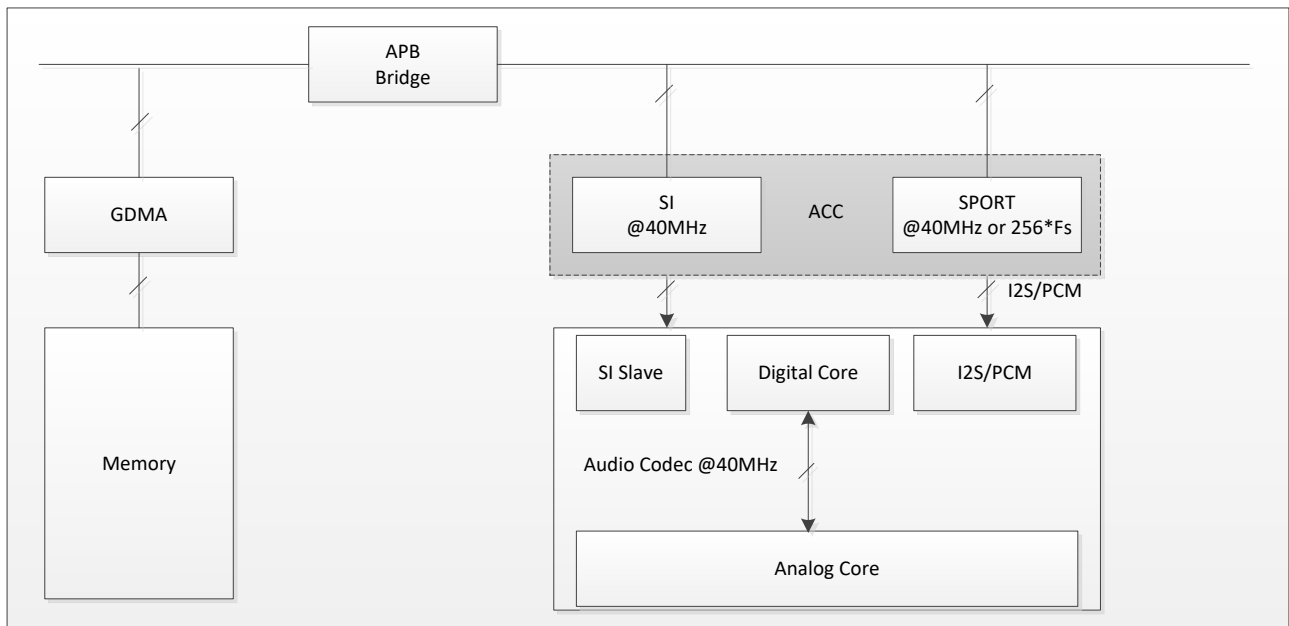


Fig 18-1 Ameba-D ACC + AC architecture

18.3.1 Block Diagram

The ACC block diagram is shown in Fig 18-2.

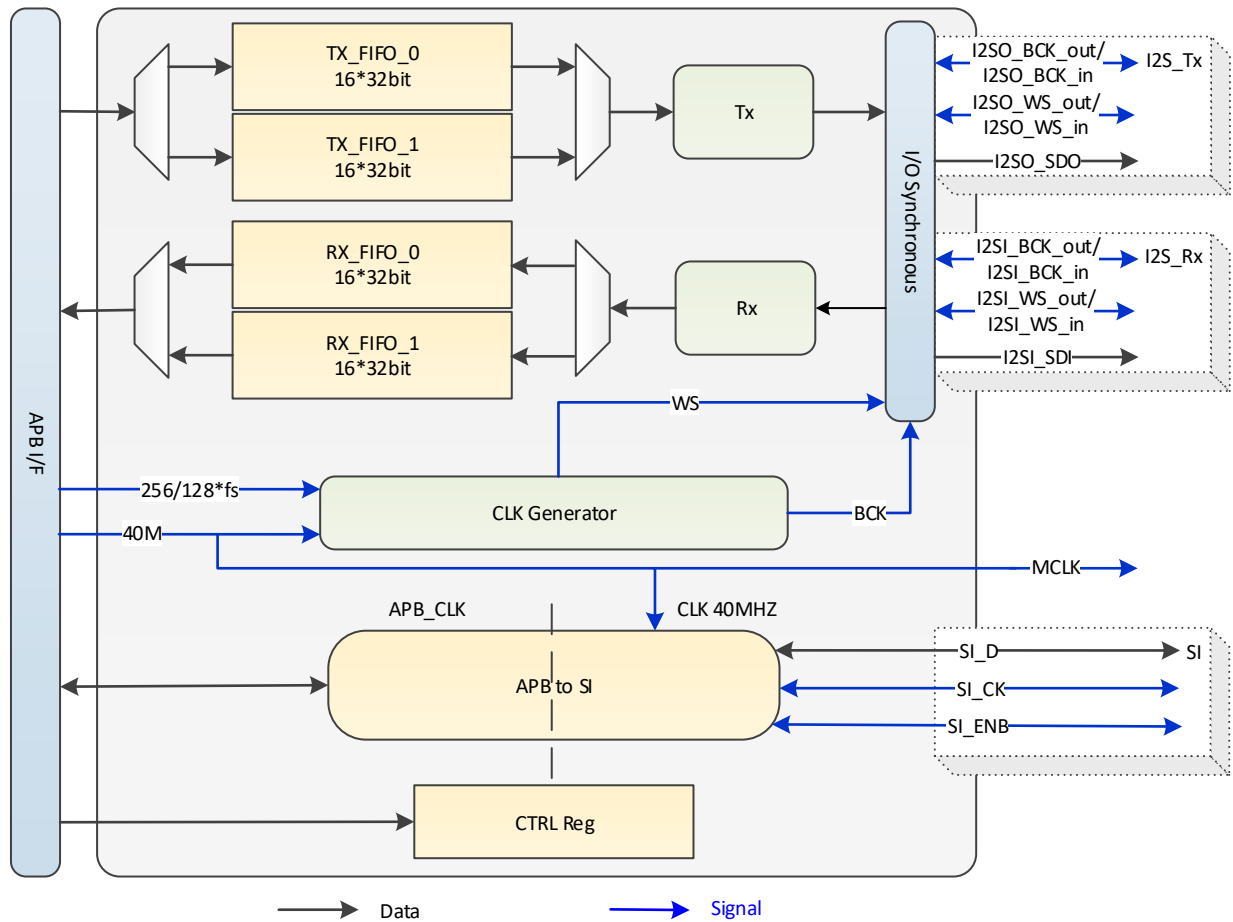


Fig 18-2 ACC block diagram

18.3.2 Data Part

Memory <-> GDMA <-> SPORT

18.3.2.1 FIFO Layout

18.3.2.1.1 16-bit Sampling Resolution

16-bit stereo data and mono data without `lr_swap/byte_swap` are listed in Table 18-1 and Table 18-2.

Table 18-1 16-bit stereo data without `lr_swap/byte_swap`

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	R ₀ [15:8]	R ₀ [7:0]	L ₀ [15:8]	L ₀ [7:0]
0x0004	R ₁ [15:8]	R ₁ [7:0]	L ₁ [15:8]	L ₁ [7:0]
0x0008	R ₂ [15:8]	R ₂ [7:0]	L ₂ [15:8]	L ₂ [7:0]
0x000C	R ₃ [15:8]	R ₃ [7:0]	L ₃ [15:8]	L ₃ [7:0]
0x0010	R ₄ [15:8]	R ₄ [7:0]	L ₄ [15:8]	L ₄ [7:0]

Table 18-2 16-bit mono data without `lr_swap/byte_swap`

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	L ₁ [15:8]	L ₁ [7:0]	L ₀ [15:8]	L ₀ [7:0]

0x0004	L ₃ [15:8]	L ₃ [7:0]	L ₂ [15:8]	L ₂ [7:0]
0x0008	L ₅ [15:8]	L ₅ [7:0]	L ₄ [15:8]	L ₄ [7:0]
0x000C	L ₇ [15:8]	L ₇ [7:0]	L ₆ [15:8]	L ₆ [7:0]
0x0010	L ₉ [15:8]	L ₉ [7:0]	L ₈ [15:8]	L ₈ [7:0]

16-bit stereo data and mono data with Ir_swap are listed in Table 18-3 and Table 18-4.

Table 18-3 16-bit stereo data with Ir_swap

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	L ₀ [15:8]	L ₀ [7:0]	R ₀ [15:8]	R ₀ [7:0]
0x0004	L ₁ [15:8]	L ₁ [7:0]	R ₁ [15:8]	R ₁ [7:0]
0x0008	L ₂ [15:8]	L ₂ [7:0]	R ₂ [15:8]	R ₂ [7:0]
0x000C	L ₃ [15:8]	L ₃ [7:0]	R ₃ [15:8]	R ₃ [7:0]
0x0010	L ₄ [15:8]	L ₄ [7:0]	R ₄ [15:8]	R ₄ [7:0]

Table 18-4 16-bit mono data with Ir_swap

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	L ₀ [15:8]	L ₀ [7:0]	L ₁ [15:8]	L ₁ [7:0]
0x0004	L ₂ [15:8]	L ₂ [7:0]	L ₃ [15:8]	L ₃ [7:0]
0x0008	L ₄ [15:8]	L ₄ [7:0]	L ₅ [15:8]	L ₅ [7:0]
0x000C	L ₆ [15:8]	L ₆ [7:0]	L ₇ [15:8]	L ₇ [7:0]
0x0010	L ₈ [15:8]	L ₈ [7:0]	L ₉ [15:8]	L ₉ [7:0]

16-bit stereo data and mono data with byte_swap are listed in Table 18-5 and Table 18-6.

Table 18-5 16-bit stereo data with byte_swap

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	R ₀ [7:0]	R ₀ [15:8]	L ₀ [7:0]	L ₀ [15:8]
0x0004	R ₁ [7:0]	R ₁ [15:8]	L ₁ [7:0]	L ₁ [15:8]
0x0008	R ₂ [7:0]	R ₂ [15:8]	L ₂ [7:0]	L ₂ [15:8]
0x000C	R ₃ [7:0]	R ₃ [15:8]	L ₃ [7:0]	L ₃ [15:8]
0x0010	R ₄ [7:0]	R ₄ [15:8]	L ₄ [7:0]	L ₄ [15:8]

Table 18-6 16-bit mono data with byte_swap

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	L ₁ [7:0]	L ₁ [15:8]	L ₀ [7:0]	L ₀ [15:8]
0x0004	L ₃ [7:0]	L ₃ [15:8]	L ₂ [7:0]	L ₂ [15:8]
0x0008	L ₅ [7:0]	L ₅ [15:8]	L ₄ [7:0]	L ₄ [15:8]
0x000C	L ₇ [7:0]	L ₇ [15:8]	L ₆ [7:0]	L ₆ [15:8]
0x0010	L ₉ [7:0]	L ₉ [15:8]	L ₈ [7:0]	L ₈ [15:8]

18.3.2.1.2 24-bit Sampling Resolution

24-bit stereo data and mono data without Ir_swap/byte_swap are listed in Table 18-7 and Table 18-8.

Table 18-7 24-bit stereo data without Ir_swap/byte_swap

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	L ₀ [23:0]			X @TX or 8'h0 @RX
0x0004	R ₀ [23:0]			X @TX or 8'h0 @RX
0x0008	L ₁ [23:0]			X @TX or 8'h0 @RX
0x000C	R ₁ [23:0]			X @TX or 8'h0 @RX
0x0010	L ₂ [23:0]			X @TX or 8'h0 @RX
0x0014	R ₂ [23:0]			X @TX or 8'h0 @RX

Table 18-8 24-bit mono data without lr_swap/byte_swap

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	L ₀ [23:0]			X @TX or 8'h0 @RX
0x0004	L ₁ [23:0]			X @TX or 8'h0 @RX
0x0008	L ₂ [23:0]			X @TX or 8'h0 @RX
0x000C	L ₃ [23:0]			X @TX or 8'h0 @RX
0x0010	L ₄ [23:0]			X @TX or 8'h0 @RX
0x0014	L ₅ [23:0]			X @TX or 8'h0 @RX

24-bit stereo data and mono data with lr_swap are listed in Table 18-9 and Table 18-10.

Table 18-9 24-bit stereo data with lr_swap

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	R ₀ [23:0]			X @TX or 8'h0 @RX
0x0004	L ₀ [23:0]			X @TX or 8'h0 @RX
0x0008	R ₁ [23:0]			X @TX or 8'h0 @RX
0x000C	L ₁ [23:0]			X @TX or 8'h0 @RX
0x0010	R ₂ [23:0]			X @TX or 8'h0 @RX
0x0014	L ₂ [23:0]			X @TX or 8'h0 @RX

Table 18-10 24-bit mono data with lr_swap

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	L ₀ [23:0]			X @TX or 8'h0 @RX
0x0004	L ₁ [23:0]			X @TX or 8'h0 @RX
0x0008	L ₂ [23:0]			X @TX or 8'h0 @RX
0x000C	L ₃ [23:0]			X @TX or 8'h0 @RX
0x0010	L ₄ [23:0]			X @TX or 8'h0 @RX
0x0014	L ₅ [23:0]			X @TX or 8'h0 @RX

24-bit stereo data and mono data with byte_swap are listed in Table 18-11 and Table 18-12.

Table 18-11 24-bit stereo data with byte_swap

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	X @TX or 8'h0 @RX			L ₀ [23:0]
0x0004	X @TX or 8'h0 @RX			R ₀ [23:0]
0x0008	X @TX or 8'h0 @RX			L ₁ [23:0]
0x000C	X @TX or 8'h0 @RX			R ₁ [23:0]
0x0010	X @TX or 8'h0 @RX			L ₂ [23:0]
0x0014	X @TX or 8'h0 @RX			R ₂ [23:0]

Table 18-12 24-bit mono data with byte_swap

Address Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x0000	X @TX or 8'h0 @RX			L ₀ [23:0]
0x0004	X @TX or 8'h0 @RX			L ₁ [23:0]
0x0008	X @TX or 8'h0 @RX			L ₂ [23:0]
0x000C	X @TX or 8'h0 @RX			L ₃ [23:0]
0x0010	X @TX or 8'h0 @RX			L ₄ [23:0]
0x0014	X @TX or 8'h0 @RX			L ₅ [23:0]

18.3.2.1.3 8-bit Sampling Resolution

8-bit mono data without lr_swap/byte_swap is listed in Table 18-13.

Table 18-13 8-bit mono data without lr_swap/byte_swap

Address Offset	[31:24]: 0x11	[23:16]: 0x10	[15:8]: 0x01	[7:0]: 0x00
----------------	---------------	---------------	--------------	-------------

0x0000	L ₃ [7:0]	L ₂ [7:0]	L ₁ [7:0]	L ₀ [7:0]
0x0004	L ₇ [7:0]	L ₆ [7:0]	L ₅ [7:0]	L ₄ [7:0]
0x0008	L ₁₁ [7:0]	L ₁₀ [7:0]	L ₉ [7:0]	L ₈ [7:0]
0x000C	L ₁₅ [7:0]	L ₁₄ [7:0]	L ₁₃ [7:0]	L ₁₂ [7:0]
0x0010	L ₁₉ [7:0]	L ₁₈ [7:0]	L ₁₇ [7:0]	L ₁₆ [7:0]

18.3.2.2 Data Exchange via I²S

The sequence of data exchange via I²S bus is based on the configuration of SPORT module, as Table 18-14 illustrates.

Table 18-14 SPORT main configuration

Data Format	I ² S	
	Left-Justified	
PCM (Long Frame Sync)	PCM (Long Frame Sync)	Mode A
		Mode B
		Mode A_N
		Mode B_N
	PCM (Short Frame Sync)	Mode A
		Mode B
		Mode A_N
		Mode B_N
Resolution	8-bit	
	16-bit (default)	
	24-bit	
Channel	Stereo (default)	
	Mono	
BCLK Polarity	BCLK (default)	
	BCLK inverse	
Serial Data	MSB first (default)	
	LSB first	
Mode	Master (default)	
	Slave	
Swap	None (default)	
	L/R Swap	
	Byte Swap	
Clock source	256*fs (default)	
	128*fs	
	40M	
Loopback	No (default)	
	SDI -> SDO	
	SDO -> SDI	

The different data formats on I²S bus are shown in Fig 18-3 to Fig 18-8.

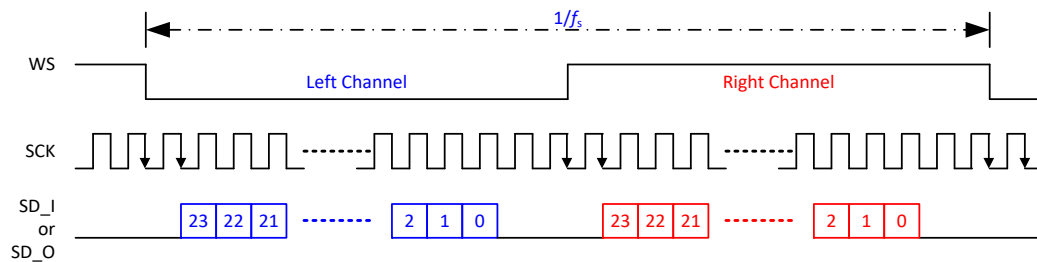


Fig 18-3 I²S audio data format

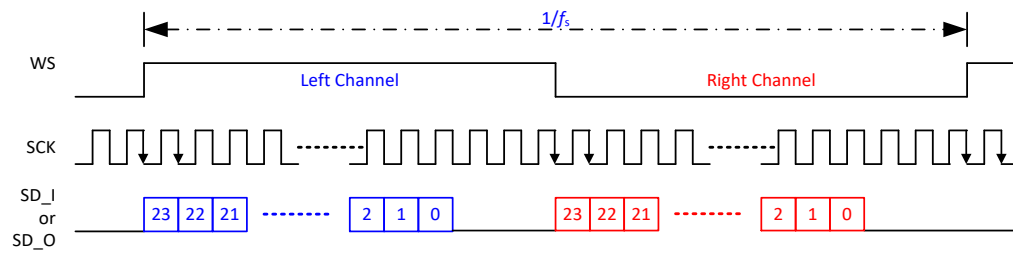


Fig 18-4 Left-Justified data format

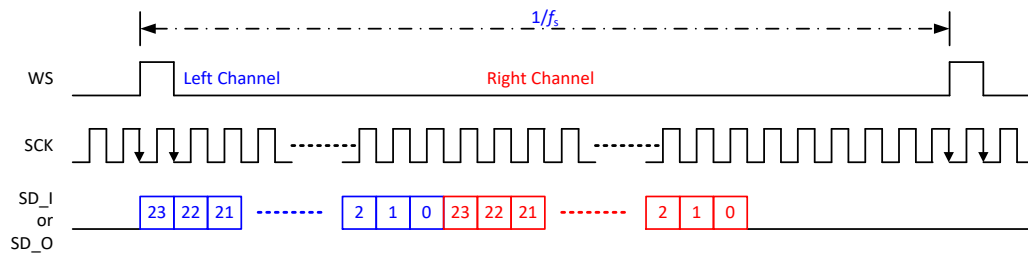


Fig 18-5 PCM mode B data format

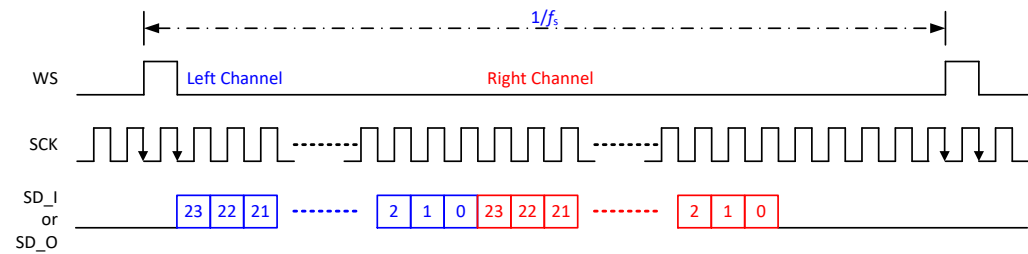


Fig 18-6 PCM mode B-N data format

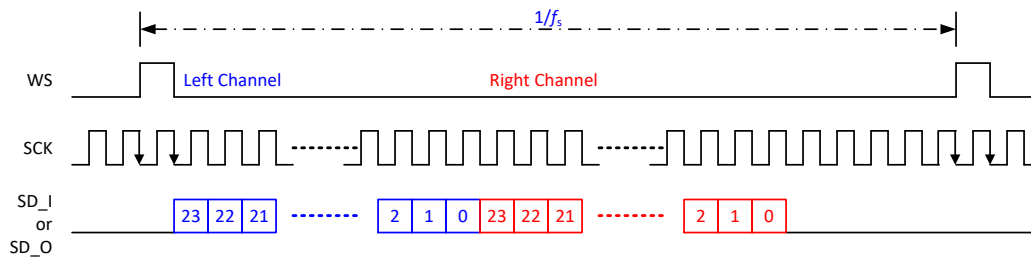


Fig 18-7 PCM mode A data format

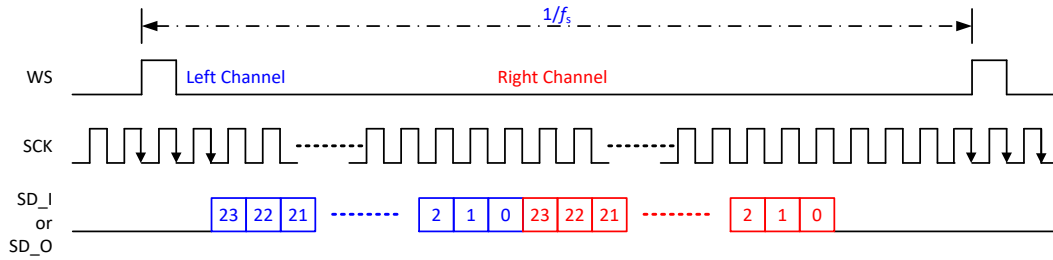


Fig 18-8 PCM mode A-N data format

18.3.3 Control Part

18.3.3.1 SPORT Control

Before transferring audio data, you need to configure the related parameters, such as sampling resolution, channel number, data format, clock source, etc.

Refer to SPORT control registers for details.

18.3.3.2 SI Control

Before transferring audio data, you also need to configure the related parameters on audio codec side. In other words, it's necessary to drive the audio codec circuit to record or play well by setting audio codec registers via SI bus.

The SI write and read timing are illustrated in Fig 18-9 and Fig 18-10 respectively.

SI write (slave side):

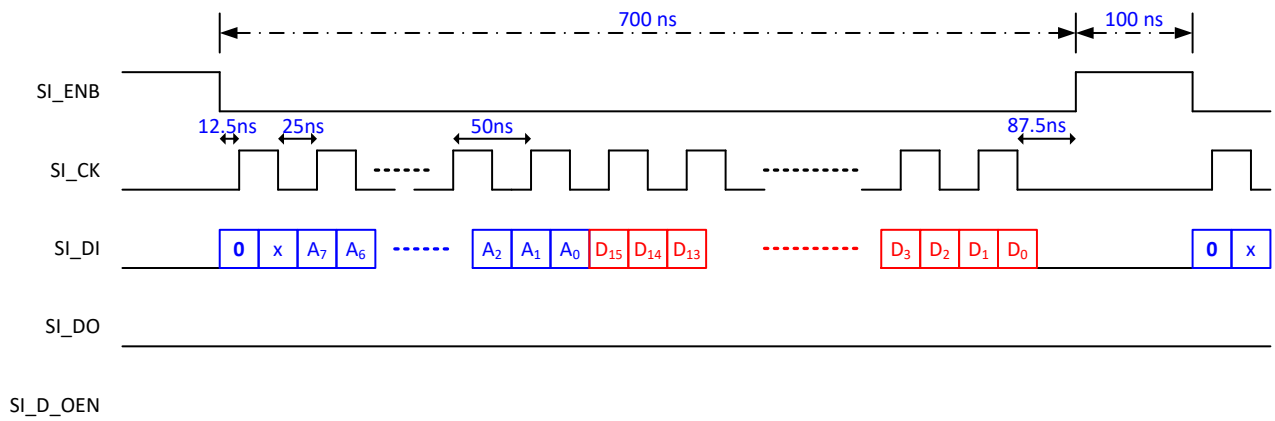


Fig 18-9 SI write timing

SI read (slave side):

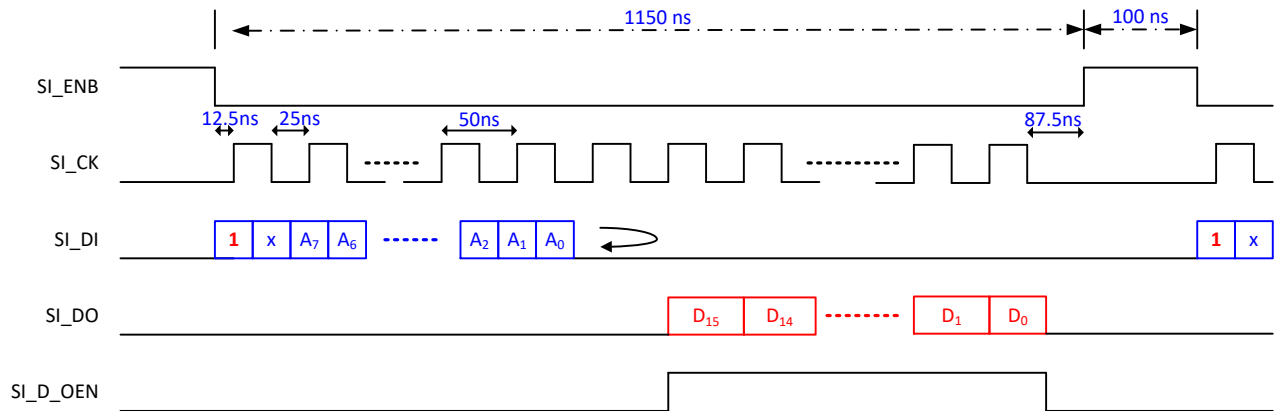


Fig 18-10 SI read timing

18.3.4 ACC Clock

The architecture of ACC clock is shown in Fig 18-11.

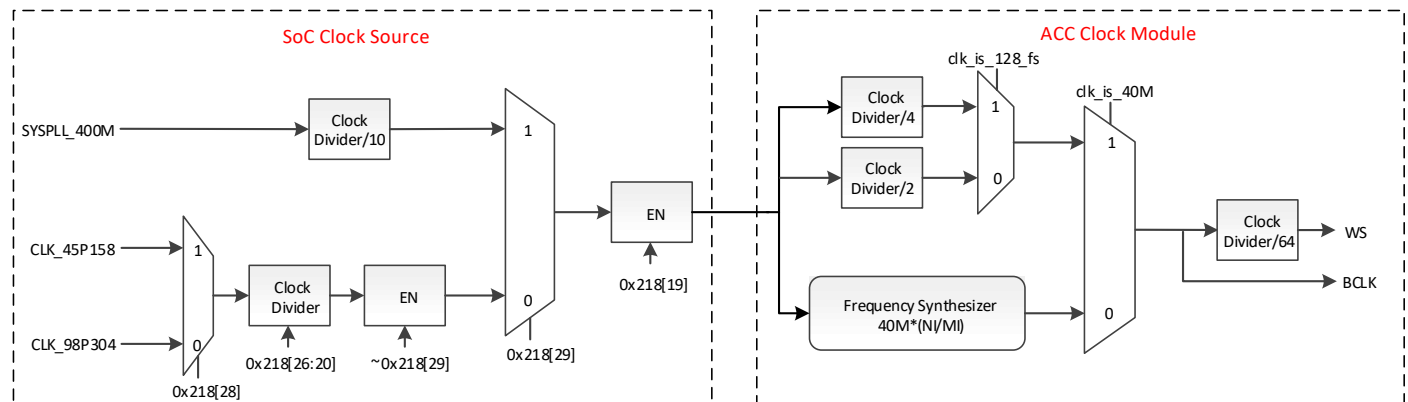


Fig 18-11 ACC clock architecture

18.4 Registers

18.4.1 SPORT Control Registers

The physical base address of SPORT control registers is 0x4001 0800.

Table 18-15 SPORT control register layout

Name	Address Offset	Access	Description
SP_TX_DR	0x0000	W	SPORT Tx data register
SP_CTRLR0	0x0004	R/W	SPORT control 0 register
SP_CTRLR1	0x0008	R/W	SPORT control 1 register
SP_CTRLR2	0x000C	R/W	SPORT control 2 register
SP_RX_DR	0x0010	R	SPORT Rx data register
SP_FIFO_SR	0x0014	R	SPORT FIFO status register
SP_ERROR_CNT_SR	0x0018	R	SPORT error counter register

SP_CLK_DIV	0x001C	R/W	SPORT frequency synthesizer register
------------	--------	-----	--------------------------------------

18.4.1.1 SP_TX_DR

- **Name:** SPORT Tx data register
- **Size:** 32 bits
- **Address offset:** 0x0000
- **Read/write access:** write

31	30	29	28	27	26	25	24	...	7	6	5	4	3	2	1	0
SP_TX_DR																
W																

Bit	Name	Access	Reset	Description
31:0	SP_TX_DR	W	32'h0	It's Tx data window between SW/GDMA and SPORT.

18.4.1.2 SP_CTRL0

- **Name:** SPORT control 0 register
- **Size:** 32 bits
- **Address offset:** 0x0004
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
LONG_FRAME_SYNC	MCLK_SEL	SP_SEL_I2S_RX_CH	SP_DAC_COMP	SP_START_RX	SP_RX_DISABLE		
R/W	R/W	R/W	R/W	R/W	R/W		
23	22	21	20	19	18	17	16
RX_LSB_FIRST	TX_LSB_FIRST	SP_SEL_I2S_TX_CH	SP_ADC_COMP	SP_START_TX	SP_TX_DISABLE		
R/W	R/W	R/W	R/W	R/W	R/W		
15	14	13	12	11	10	9	8
SP_I2S_SELF_LPB_K_EN	SP_INV_I2S_SCLK	SP_DATA_LEN_SEL	SP_EN_I2S_MONO	SP_EN_PCM_N_MODE	SP_DATA_FORMAT_SEL		
R/W	R/W	R/W	R/W	R/W	R/W		
7	6	5	4	3	2	1	0
DSP_CTL_MODE	SP_LOOPBACK	SP_WCLK_INVERSE	SLAVE_DATA_SEL	SLAVE_CLK_SEL	RX_INV_I2S_SCLK	TX_INV_I2S_SCLK	SP_RESET
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31	LONG_FRAME_SYNC	R/W	1'b0	<ul style="list-style-type: none"> ● 1'b0: Short frame sync ● 1'b1: Long frame sync
30	MCLK_SEL	R/W	1'b0	<ul style="list-style-type: none"> ● 1'b1: MCLK output=128*fs ● 1'b0: MCLK output=256*fs
29:28	SP_SEL_I2S_RX_CH	R/W	2'b00	<ul style="list-style-type: none"> ● 2'b00: L/R ● 2'b01: R/L ● 2'b10: L/L ● 2'b11: R/R @ ADC path
27:26	SP_DAC_COMP	R/W	2'b00	<ul style="list-style-type: none"> ● 2'b00: OFF ● 2'b01: u law ● 2'b10: A law
25	SP_START_RX	R/W	1'b0	<ul style="list-style-type: none"> ● 1'b0: Rx is disabled ● 1'b1: Rx is started
24	SP_RX_DISABLE	R/W	1'b1	<ul style="list-style-type: none"> ● 1'b1: SPORT Rx is disabled ● 1'b0: SPORT Rx is enabled
23	RX_LSB_FIRST	R/W	1'b0	<ul style="list-style-type: none"> ● 1'b0: MSB first when Rx ● 1'b1: LSB first
22	TX_LSB_FIRST	R/W	1'b0	<ul style="list-style-type: none"> ● 1'b0: MSB first when Tx

				<ul style="list-style-type: none"> 1'b1: LSB first
21:20	SP_SEL_I2S_TX_CH	R/W	2'b00	<ul style="list-style-type: none"> 2'b00: L/R 2'b01: R/L 2'b10: L/L 2'b11: R/R @ DAC path
19:18	SP_ADC_COMP	R/W	2'b00	<ul style="list-style-type: none"> 2'b00: OFF 2'b01: u law 2'b10: A law
17	SP_START_TX	R/W	1'b0	<ul style="list-style-type: none"> 1'b0: Tx is disabled 1'b1: Tx is started
16	SP_TX_DISABLE	R/W	1'b1	<ul style="list-style-type: none"> 1'b1: SPORT TX is disabled 1'b0: SPORT TX is enabled
15	SP_I2S_SELF_LPBK_EN	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: Internal loopback mode is enabled 1'b0: Internal loopback mode is disabled
14	SP_INV_I2S_SCLK	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: I2S/PCM bit clock is inverted 1'b0: I2S/PCM bit clock is not inverted
13:12	SP_DATA_LEN_SEL	R/W	2'b00	<ul style="list-style-type: none"> 2'b00: 16 bits 2'b10: 24 bits 2'b11: 8 bits
11	SP_EN_I2S_MONO	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: Mono 1'b0: Stereo
10	SP_EN_PCM_N_MODE	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: PCM negative mode 1'b0: PCM normal mode
9:8	SP_DATA_FORMAT_SEL	R/W	2'b00	<ul style="list-style-type: none"> 3'b000: I2S 3'b001: Left-Justified 3'b010: PCM mode A 3'b011: PCM mode B 3'b110: PCM mode A-N 3'b111: PCM mode B-N
7	DSP_CTL_MODE	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: DSP and SPORT handshaking is enabled. 1'b0: GDMA and SPORT handshaking is enabled
6	SP_LOOPBACK	R/W	1'b0	1'b1: Self loopback mode
5	SP_WCLK_INVERSE	R/W	1'b0	1'b1: I2S/PCM word clock is inverted
4	SLAVE_DATA_SEL	R/W	1'b0	1'b1: To be an I2S or PCM slave (data path)
3	SLAVE_CLK_SEL	R/W	1'b0	1'b1: To be an I2S or PCM slave (CLK path)
2	RX_INV_I2S_SCLK	R/W	1'b0	1'b1: SCLK to Rx path (ADC path) is inverted.
1	TX_INV_I2S_SCLK	R/W	1'b0	1'b1: SCLK to Tx path (DAC path) is inverted.
0	SP_RESET	R/W	1'b0	1'b1: Reset SPORT module, remember to write 1 first and then write 0.

18.4.1.3 SP_CTRLR1

- **Name:** SPORT control 1 register
- **Size:** 32 bits
- **Address offset:** 0x0008
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
INT_ENABLE							
R/W							
23	22	21	20	19	18	17	16
RX_SNK_LR_SW AP	RX_SNK_BYTE_S WAP	TX_SRC_LR_S WAP	TX_SRC_BYTE_S WAP	RSVD		MODE_128FS	MODE_40MHZ
R/W	R/W	R/W	R/W			R/W	R/W
15	14	13	12	11	10	9	8
RSVD		CLEAR_RX_ER R_CNT	CLEAR_TX_ERR_ CNT	RSVD		DEBUG_BUS_SEL	
		R/W	R/W			R/W	
7	6	5	4	3	2	1	0

FRAME_SYNC_OFFSET
R/W

Bit	Name	Access	Reset	Description
31:24	INT_ENABLE	R/W	8'h0	<ul style="list-style-type: none"> 0: For the interrupt of 'SP_READY_TO_TX' 1: For the interrupt of 'SP_READY_TO_RX'
23	RX_SNK_LR_SWAP	R/W	1'b0	1'b1: Swap L/R audio samples written to the sink memory
22	RX_SNK_BYTE_SWAP	R/W	1'b0	1'b1: Swap H/L bytes written to the sink memory
21	TX_SRC_LR_SWAP	R/W	1'b0	1'b1: Swap L/R audio samples read from the source memory
20	TX_SRC_BYTE_SWAP	R/W	1'b0	1'b1: Swap H/L bytes read from the source memory
19:18	RSVD	N/A	6'h0	Reserved
17	MODE_128FS	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: Clock source is 128*fs 1'b0: Clock source is 256*fs
16	MODE_40MHZ	R/W	1'b0	<ul style="list-style-type: none"> 1'b1: Clock source is 40MHz 1'b0: Clock source is 128*fs or 256*fs
15:14	RSVD	N/A	2'b00	Reserved
13	CLEAR_RX_ERR_CNT	R/W	1'b0	Write 1'b1 and then write 1'b0 to clear Rx error counter.
12	CLEAR_TX_ERR_CNT	R/W	1'b0	Write 1'b1 and then write 1'b0 to clear Tx error counter.
11	RSVD	N/A	1'b0	Reserved
10:8	DEBUG_BUS_SEL	R/W	3'h0	<ul style="list-style-type: none"> 3'b000: debug_bus_a 3'b001: debug_bus_b 3'b111: debug_bus_h
7:0	FRAME_SYNC_OFFSET	R/W	8'h81	To control the length of 'long_frame_sync' signal when it is ON. It is set to 50% of the period of the sampling rate by default.

18.4.1.4 SP_CTRLR2

- **Name:** SPORT control 2 register
- **Size:** 32 bits
- **Address offset:** 0x000C
- **Read/write access:** read/write

31	30	29	...	8	7	6
RSVD						
5	4	3	2	1	0	
RX_FIFO_DEPTH_HALF_SEL	TX_FIFO_DEPTH_HALF_SEL	RSVD				
R/W	R/W					

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	26'h0	Reserved
5	RX_FIFO_DEPTH_HALF_SEL	R/W	1'b0	1'b1: Rx FIFO depth will reduce to half. It can reduce the data path latency.
4	TX_FIFO_DEPTH_HALF_SEL	R/W	1'b0	1'b1: Tx FIFO depth will reduce to half. It can reduce the data path latency.
3:0	RSVD	N/A	4'b0	Reserved

18.4.1.5 SP_RX_DR

- **Name:** SPORT Rx data register
- **Size:** 32 bits
- **Address offset:** 0x0010
- **Read/write access:** read

31	30	29	28	27	26	25	24	...	7	6	5	4	3	2	1	0
SP_RX_DR																
WR																

Bit	Name	Access	Reset	Description
31:0	SP_RX_DR	R	32'h0	It's Rx data window between SW/GDMA and SPORT.

18.4.1.6 SP_FIFO_SR

- **Name:** SPORT FIFO status register
- **Size:** 32 bits
- **Address offset:** 0x0014
- **Read/write access:** read

31	30	29	28	27	26	25	24
RX1_RCNT_BUS							
R							
23	22	21	20	19	18	17	16
RX0_RCNT_BUS							
R							
15	14	13	12	11	10	9	8
TX1_WCNT_BUS							
R							
7	6	5	4	3	2	1	0
TX0_WCNT_BUS							
R							

Bit	Name	Access	Reset	Description
31:24	RX1_RCNT_BUS	R	8'h0	Rx1 FIFO read counter status (MIC path)
23:16	RX0_RCNT_BUS	R	8'h0	Rx0 FIFO read counter status (MIC path)
15:8	TX1_WCNT_BUS	R	8'h0	Tx1 FIFO write counter status (SPK path)
7:0	TX0_WCNT_BUS	R	8'h0	Tx0 FIFO write counter status (SPK path)

18.4.1.7 SP_ERROR_CNT_SR

- **Name:** SPORT error counter register
- **Size:** 32 bits
- **Address offset:** 0x0018
- **Read/write access:** read

31	30	29	28	27	26	25	24
RSVD							
23	22	21	20	19	18	17	16
RSVD							
15	14	13	12	11	10	9	8
RX_ERR_CNT							
R							
7	6	5	4	3	2	1	0
TX_ERR_CNT							
R							

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	16'h0	Reserved
15:8	RX_ERR_CNT	R	8'h0	Rx error counter (MIC path) Note: This counter should always be zero if everything works well.
7:0	TX_ERR_CNT	R	8'h0	Tx error counter (SPK path) Note: This counter should always be zero if everything works well.

18.4.1.8 SP_CLK_DIV

- **Name:** SPORT frequency synthesizer register
- **Size:** 32 bits
- **Address offset:** 0x001C
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MI_NI_UPDATE								NI							
R/W								R/W							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MI															
R/W															

Bit	Name	Access	Reset	Description
31	MI_NI_UPDATE	R/W	1'b0	1'b1: to update 'MI' and 'NI' to get the new clock rate. This bit is reset automatically when the update is done.
30:16	NI	R/W	15'd48	BCLK = 40MHz * (NI/MI)
15:0	MI	R/W	16'd625	For example: BCLK = 3.072MHz = 40MHz * (48/625)

18.4.2 SI Control Registers

The physical base address of SI control registers is 0x4001 0000.

Table 18-16 SI control register layout

Name	Address Offset	Access	Description
SI_CTRLR0	0x0000	R/W	SI control 0 register
SI_CLK_EN	0x0004	R/W	Audio codec clock control register

18.4.2.1 SI_CTRLR0

- **Name:** SI control 0 register
- **Size:** 32 bits
- **Address offset:** 0x0000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SI_DATA															
R/W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SI_ADDRESS								SI_DISABLE		RSVD		SI_READ_START		RSVD	
R/W								R/W		R/W		R/W		R/W	

Bit	Name	Access	Reset	Description
31:16	SI_DATA	R/W	0	<ul style="list-style-type: none"> ● For SI_WRITE, this field is 16-bit data written to the specified address in AC. ● For SI_READ, this field is 16-bit data read from the specified address in AC.
15:8	SI_ADDRESS	R/W	8'h0	8-bit address to audio codec
7	SI_DISABLE	R/W	1'b0	1'b1: SI read/write function is not enabled
6:5	RSVD	R/W	2'h0	Reserved
4	SI_READ_START	R/W	1'b0	1'b1: Start to perform SI read to audio codec This bit is cleared when SI_READ is done.
3:1	RSVD	N/A	3'h0	Reserved
0	SI_WR_START	R/W	1'b0	1'b1: Start to perform SI write to audio codec This bit is cleared when SI_WRITE is done.

18.4.2.2 SI_CLK_EN

- **Name:** Audio codec clock control register
- **Size:** 32 bits
- **Address offset:** 0x0004
- **Read/write access:** read/write

31	30	29	28	27	26	25	...	7	6	5	4	3	2	1	0
RSVD															REG_CLK_EN
															R/W

Bit	Field	Access	Reset	Description
31:1	RSVD	N/A	0	Reserved
0	REG_CLK_EN	R/W	1	<ul style="list-style-type: none"> ● 1'b1: Turn on the clock of the register bank of audio codec. ● 1'b0: Turn off the clock. To save power, reset this bit to gate clock when the registers are already programmed.

19 Serial Peripheral Interface (SPI)

19.1 Product Overview

Ameba-D support Motorola Serial Peripheral Interface (SPI) – A four-wire, full-duplex serial protocol from Motorola.

19.1.1 Block Diagram

Fig 19-1 shows the following functional groupings of the main interfaces to the SPI block.

- APB interface and DMA Controller Interface
- Transmit and receive FIFO controllers and an FSM controller
- Register block
- Shift control and interrupt logic

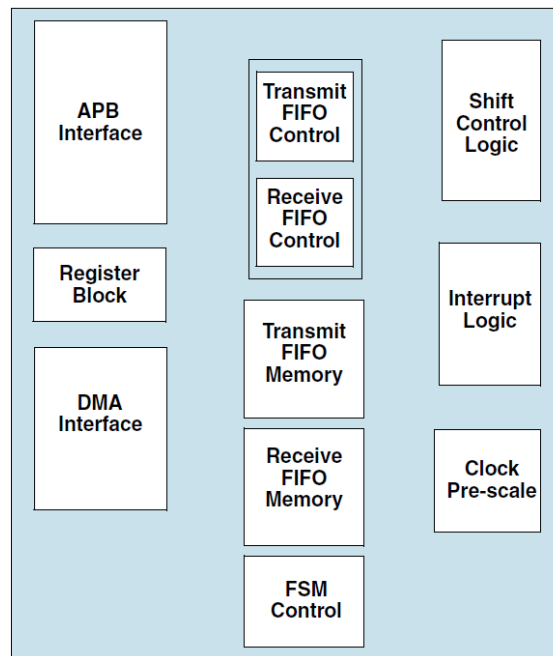


Fig 19-1 SPI block diagram

19.1.2 Features

Ameba-D SPI has the following features:

- Support Motorola SPI Serial interface operation
- Support master or slave operation mode
- Provide two SPI ports
 - SPI0 (High speed): configured as master or slave with Max. Baud rate: 50MHz.
 - SPI1 (Normal speed): configured as master with Max. Baud rate: 25MHz.
- Support DMA interface for DMA transfer
- Independent masking of interrupts
- FIFO depth – The transmit and receive FIFO buffers 64 words deep. The FIFO width is fixed at 16 bits.
- Hardware/software slave-select – Dedicated hardware slave-select lines can be used or software control can be used to target the serial-slave device
- Programmable features
 - Clock bit-rate – Dynamic control of the serial bit rate of the data transfer; used in only serial-master mode of operation.

- Data item size (4 to 16 bits) – Item size of each data transfer under the control of the programmer.
- Configurable clock polarity and phase
- Programmable delay on the sample time of the received serial data bit (rxd), when configured in Master Mode; enables programmable control of routing delays resulting in higher serial data-bit rates.

19.2 Functional Description

This chapter describes the functional operation of the SPI.

The SPI is a configurable, synthesizable, and programmable component that is a full-duplex master or slave-synchronous serial interface. It can be configured in one of two modes of operations: as a serial master or a serial slave. The SPI can connect to any serial-master or serial-slave peripheral device using Motorola Serial Peripheral Interface (SPI).

19.2.1 Overview

In order to connect to a serial-master or serial-slave peripheral device for the SPI, the peripheral must have Motorola Serial Peripheral Interface (SPI) – A four-wire, full-duplex serial protocol from Motorola.

The serial protocols supported by the SPI allow for serial slaves to be selected or addressed using either hardware or software. When implemented in hardware, only one slave can be selected under the control of hardware select line. When implemented in software, users can use GPIO to control more SPI slaves. And there cannot be more than one slave be selected at any time. In this mode, it is assumed that the serial master has only a single slave select output.

19.2.1.1 Motorola Serial Peripheral Interface (SPI)

There are four possible combinations for the serial clock phase and clock polarity. The clock phase (SCPH) determines whether the serial transfer begins with the falling edge of the slave select signal or the first edge of the serial clock. The clock polarity (SCPOL) configuration parameter determines whether the inactive state of the serial clock is high or low. To transmit data, both SPI peripherals must have identical serial clock phase (SCPH) and clock polarity (SCPOL) values.

When the configuration parameter SCPH = 0, data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the txd and rxd lines prior to the first serial clock edge. Fig 19-2 shows a timing diagram for a single SPI data transfer with SCPH = 0. The serial clock is shown for configuration parameters SCPOL = 0 and SCPOL = 1.

The signals are illustrated in the timing diagrams in this section:

- sclk_out – serial clock from SPI master (master configuration only)
- sclk_in – serial clock from SPI slave (slave configuration only)
- ss_0_n – slave select signal from SPI master (master configuration only)
- ss_in_n – slave select input to the SPI slave
- ss_oe_n – output enable for the SPI master/slave
- txd – transmit data line for the SPI master/slave
- rxd – receive data line for the SPI master/slave

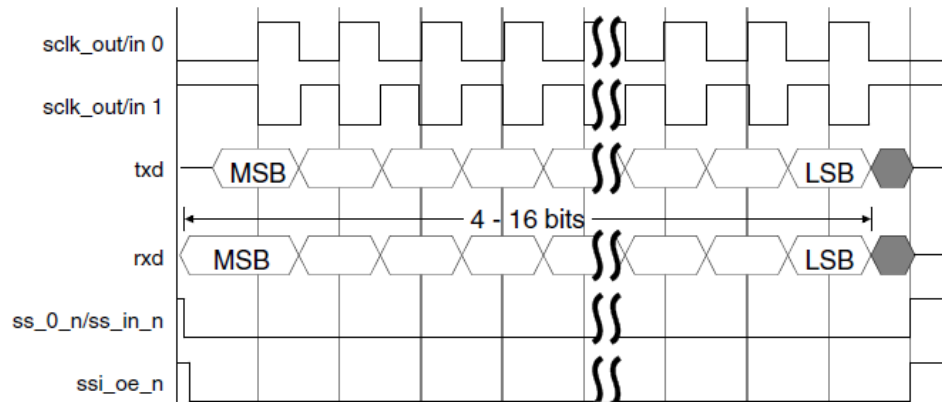


Fig 19-2 SPI Serial Format (SCPH = 0)

As data transmission starts on the falling edge of the slave select signal when SCPH = 0, whether the slave select signal of SPI master should toggle or not before the next data frame during continuous data transfer can be configured through SS_T in CTRLR0 register. Ameba-D SPI slave can communicate with serial-master no matter the slave select signal from master is toggling or not during continuous data transfer. The timing diagram of SS toggling is illustrated in Fig 19-3. The timing diagram of SS not-toggling is illustrated in Fig 19-4.

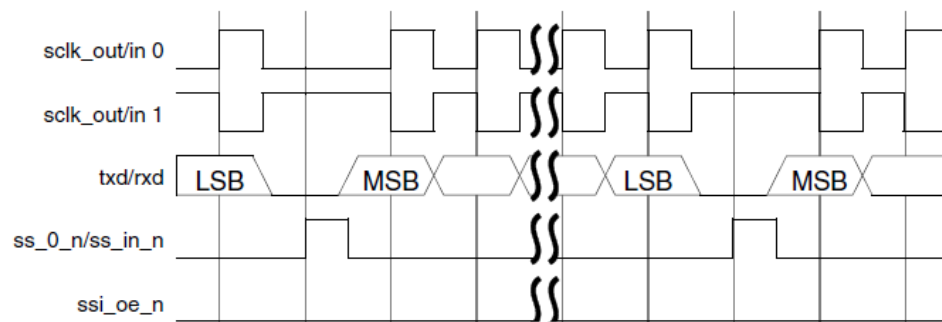


Fig 19-3 SPI Serial Format Continuous Transfers (SCPH = 0 and SS toggling)

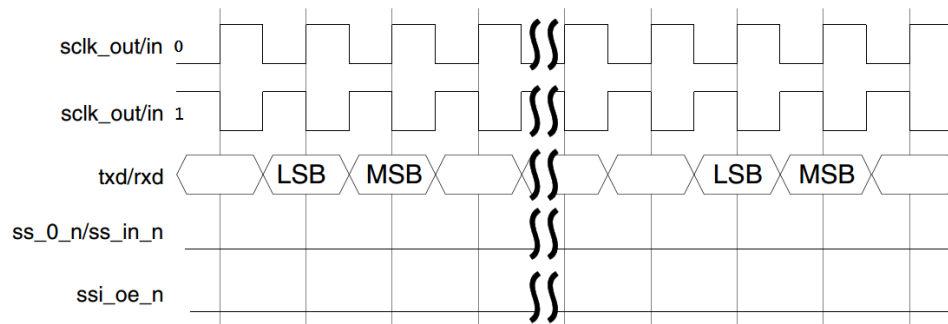


Fig 19-4 SPI Serial Format Continuous Transfers (SCPH = 0 and SS not-toggling)

When the configuration parameter SCPH = 1, both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the master and slave peripherals on the leading edge of the serial clock. During continuous data frame transfers, the slave select line may be held active-low until the last bit of the last frame has been captured. Fig 19-5 shows the timing diagram for the SPI format when the configuration parameter SCPH = 1.

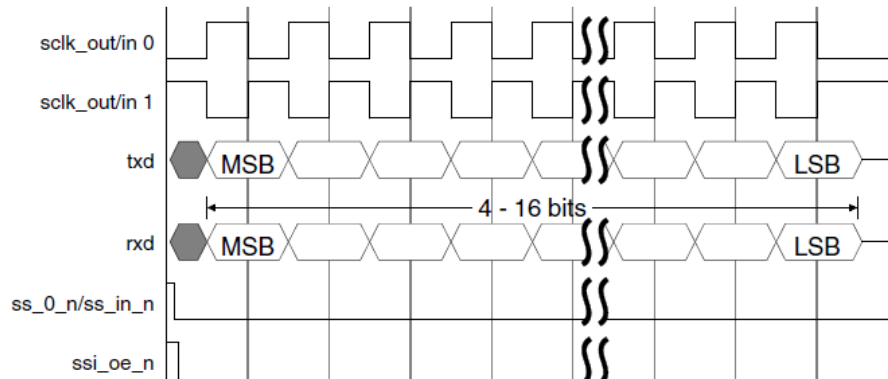


Fig 19-5 SPI Serial Format (SCPH = 1)

Continuous data frames are transferred in the same way as single frames, with the MSB of the next frame following directly after the LSB of the current frame. The slave select signal is held active for the duration of the transfer. Fig 19-6 shows the timing diagram for continuous SPI transfers when the configuration parameter SCPH = 1.

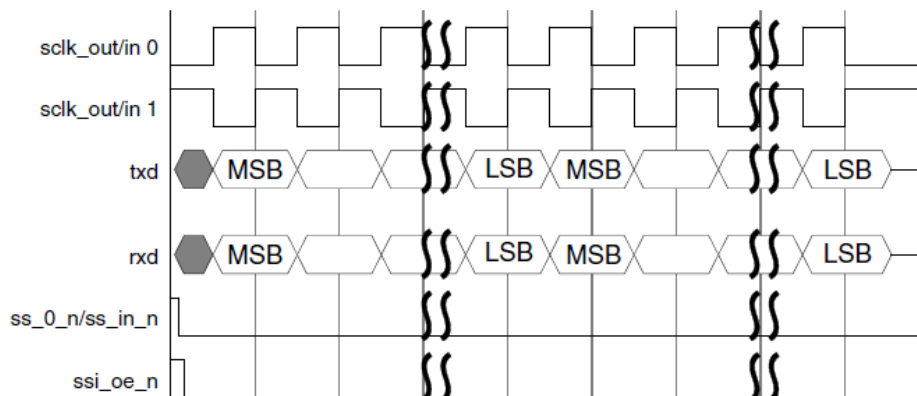


Fig 19-6 SPI Serial Format Continuous Transfers (SCPH = 1)

19.2.1.2 Clock Ratios

When SPI is configured as a master device, the maximum frequency of the bit-rate clock (sclk_out) is one-half the frequency of ssi_clk. This allows the shift control logic to capture data on one clock edge of sclk_out and propagate data on the opposite edge; this is illustrated in Fig 19-7. The sclk_out line toggles only when an active transfer is in progress. At all other times it is held in an inactive state, as defined by the serial protocol under which it operates.

The frequency of sclk_out can be derived from the following equation:

$$F_{sclk_out} = \frac{F_{ssi_clk}}{SCKDV}$$

SCKDV is a bit field in the programmable register BAUDR, holding any even value in the range 0 to 65,534. If SCKDV is 0, then sclk_out is disabled.

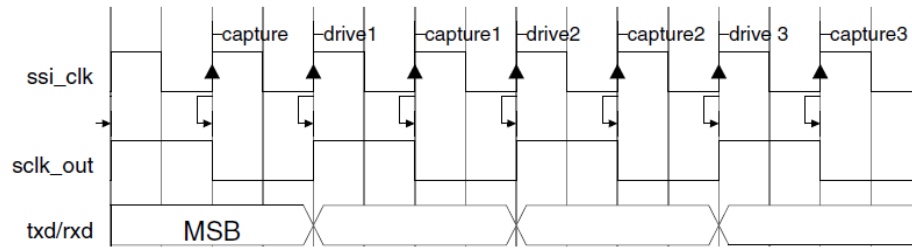


Fig 19-7 Maximum sclk_out/ssi_clk Ratio

Ameba-D support high speed SPI slave. When SPI is configured as a slave device, the maximum frequency of the bit-rate clock (sclk_in) provided by serial master can reach up to one-half the frequency of ssi_clk.

A summary of the frequency ratio restrictions between the bit-rate clock (sclk_out/sclk_in) and the SPI peripheral clock (ssi_clk) are described as:

- Master: $F_{ssi_clk} \geq 2 \times (\text{maximum } F_{sclk_out})$
- Slave: $F_{ssi_clk} \geq 2 \times (\text{maximum } F_{sclk_in})$

The F_{ssi_clk} of SPI0 is 100MHz. The F_{ssi_clk} of SPI1 is 50MHz.

19.2.1.3 Transmit and Receive FIFO Buffers

The depth of both transmit and receive FIFO buffers is 64. The width is fixed at 16 bits due to the serial specifications, which state that a serial transfer (data frame) can be 4 to 16 bits in length. Each data entry in the FIFO buffers contains a single data frame.

Note: The transmit and receive FIFO buffers are cleared when the SPI is disabled (SSI_EN = 0) or when it is reset.

The transmit FIFO is loaded by APB write commands to the SPI data register (DR). Data are popped (removed) from the transmit FIFO by the shift control logic into the transmit shift register. The transmit FIFO generates a FIFO empty interrupt request (ssi_txe_intr) when the number of entries in the FIFO is less than or equal to the FIFO threshold value. The threshold value, set through the programmable register TXFTLR, determines the level of FIFO entries at which an interrupt is generated. The threshold value allows you to provide early indication to the processor that the transmit FIFO is nearly empty. A transmit FIFO overflow interrupt (ssi_txo_intr) is generated if you attempt to write data into an already full transmit FIFO.

Data are popped from the receive FIFO by APB read commands to the SPI data register (DR). The receive FIFO is loaded from the receive shift register by the shift control logic. The receive FIFO generates a FIFO-full interrupt request (ssi_rxf_intr) when the number of entries in the FIFO is greater than or equal to the FIFO threshold value plus 1. The threshold value, set through programmable register RXFTLR, determines the level of FIFO entries at which an interrupt is generated.

The threshold value allows you to provide early indication to the processor that the receive FIFO is nearly full. A receive FIFO overrun interrupt (ssi_rxo_intr) is generated when the receive shift logic attempts to load data into a completely full receive FIFO. However, this newly received data are lost. A receive FIFO underflow interrupt (ssi_rxu_intr) is generated if you attempt to read from an empty receive FIFO. This alerts the processor that the read data are invalid.

19.2.1.4 SPI Interrupts

The SPI supports combined and individual interrupt requests, each of which can be masked. The combined interrupt request is the ored result of all other SPI interrupts after masking. The system designer has the choice of routing individual interrupt requests or only the combined interrupt request to the Interrupt Controller. The SPI interrupts are described as follows:

- Transmit FIFO Empty Interrupt (ssi_txe_intr) – Set when the transmit FIFO is equal to or below its threshold value and requires service to prevent an under-run. The threshold value, set through a software-programmable register, determines the level of transmit FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are written into the transmit FIFO buffer, bringing it over the threshold level.
- Transmit FIFO Overflow Interrupt (ssi_txo_intr) – Set when an APB access attempts to write into the transmit FIFO after it has been completely filled. When set, data written from the APB is discarded. This interrupt remains set until you read the transmit FIFO overflow interrupt clear register (TXOICR).

- Receive FIFO Full Interrupt (`ssi_rxf_intr`) – Set when the receive FIFO is equal to or above its threshold value plus 1 and requires service to prevent an overflow. The threshold value, set through a software-programmable register, determines the level of receive FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are read from the receive FIFO buffer, bringing it below the threshold level.
- Receive FIFO Overflow Interrupt (`ssi_rxo_intr`) – Set when the receive logic attempts to place data into the receive FIFO after it has been completely filled. When set, newly received data are discarded. This interrupt remains set until you read the receive FIFO overflow interrupt clear register (`RXOICR`).
- Receive FIFO Underflow Interrupt (`ssi_rxu_intr`) – Set when an APB access attempts to read from the receive FIFO when it is empty. When set, zeros are read back from the receive FIFO. This interrupt remains set until you read the receive FIFO underflow interrupt clear register (`RXUICR`).
- Multi-Master Contention Interrupt (`ssi_mst_intr`) – Present only when the SPI component is configured as a serial-master device. The interrupt is set when another serial master on the serial bus selects the SPI master as a serial-slave device and is actively transferring data. This informs the processor of possible contention on the serial bus. This interrupt remains set until you read the multi-master interrupt clear register (`MSTICR`).
- Combined Interrupt Request (`ssi_intr`) – OR'ed result of all the above interrupt requests after masking. To mask this interrupt signal, you must mask all other SPI interrupt requests.

19.2.2 Transfer Modes

When transferring data on the serial bus, the SPI master can operate in transmit and receive mode, transmit only mode or receive only mode as following. The transfer mode (TMOD) is set by writing to control register 0 (`CTRLR0`).

When transferring data on the serial bus, the SPI slave can only operate in transmit and receive mode. That is, TMOD field in `CTRLR0` register is invalid for SPI slave. If you do not want this device to respond with data, slave output can be disabled through `SLV_OE` bit in `CTRLR0`.

19.2.2.1 Transmit and Receive

When `TMOD = 2'b00`, both transmit and receive logic are valid. Transmit data are popped from the transmit FIFO and sent through the `txd` line to the target device, which replies with data on the `rx` line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

19.2.2.2 Transmit Only

When `TMOD = 2'b01`, the receive data are invalid and should not be stored in the receive FIFO. Transmit data are popped from the transmit FIFO and sent through the `txd` line to the target device, which replies with data on the `rx` line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO. The data in the receive shift register is overwritten by the next transfer. You should mask interrupts originating from the receive logic when this mode is entered.

19.2.2.3 Receive Only

When `TMOD = 2'b10`, the transmit data are invalid. The `txd` output remains at a constant logic level during the transmission. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. You should mask interrupts originating from the transmit logic when this mode is entered.

19.2.3 Operation Modes

The SPI can be configured in the fundamental modes of operation discussed in this section.

19.2.3.1 Serial-Master Mode

This mode enables serial communication with serial-slave peripheral devices. When configured as a serial-master device, the SPI initiates and controls all serial transfers. Fig 19-8 shows an example of the SPI configured as a serial master with all other devices on the serial bus configured as serial slaves.

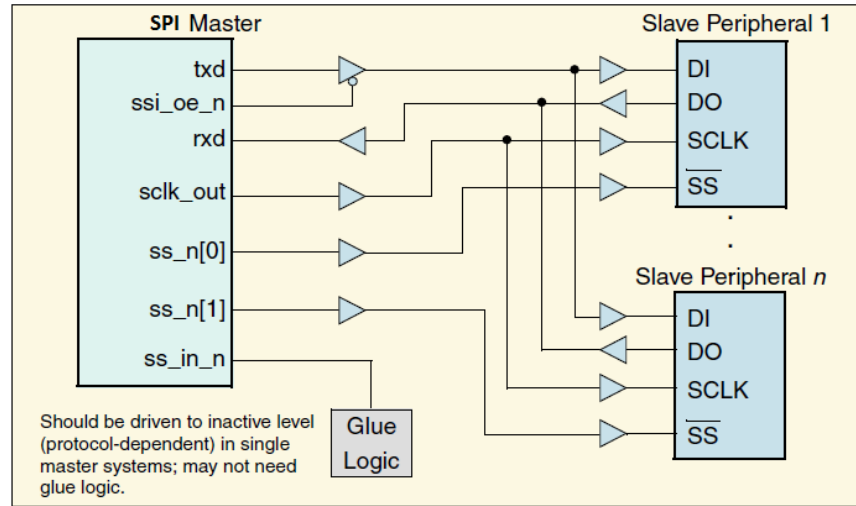


Fig 19-8 SPI Configured as master device

The serial bit-rate clock, generated and controlled by the SPI, is driven out on the sclk_out line. When the SPI is disabled (SSI_EN = 0), no serial transfers can occur and sclk_out is held in inactive state.

19.2.3.1.1 RXD Sample Delay

When the SPI is configured as a master, additional logic can be included in the design in order to delay the default sample time of the rxd signal. This additional logic can help to increase the maximum achievable frequency on the serial bus.

Round trip routing delays on the sclk_out signal from the master and the rxd signal from the slave can mean that the timing of the rxd signal—as seen by the master—has moved away from the normal sampling time. Fig 19-9 illustrates this situation.

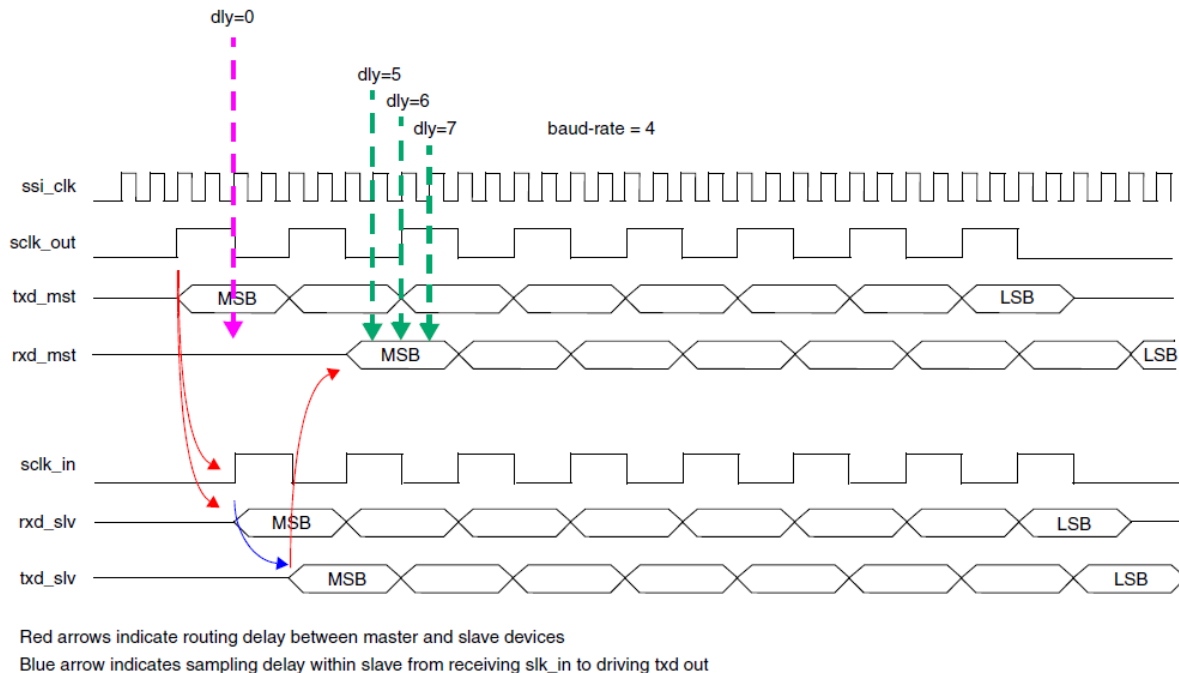


Fig 19-9 Effects of round trip routing delays on sclk_out signal

The Slave uses the `sclk_out` signal from the master as a strobe in order to drive `rx_d` signal data onto the serial bus. Routing and sampling delays on the `sclk_out` signal by the slave device can mean that the `rx_d` bit has not stabilized to the correct value before the master samples the `rx_d` signal. Fig 19-9 shows an example of how a routing delay on the `rx_d` signal can result in an incorrect `rx_d` value at the default time when the master samples the port.

When the RXD Sample Delay logic is included, the user can dynamically program a delay value in order to move the sampling time of the `rx_d` signal equal to a number of `ssi_clk` cycles from the default.

The sample delay logic has a resolution of one `ssi_clk` cycle. Software can “train” the serial bus by coding a loop that continually reads from the slave and increments the master's RXD Sample Delay value until the correct data is received by the master.

19.2.3.1.2 Data Transfers

Data transfers are started by the serial-master device. When the SPI is enabled (`SSI_EN=1`), at least one valid data entry is present in the transmit FIFO and a serial-slave device is selected. When actively transferring data, the busy flag (BUSY) in the status register ([SR](#)) is set. You must wait until the busy flag is cleared before attempting a new serial transfer.

Note: The BUSY status is not set when the data are written into the transmit FIFO. This bit gets set only when the target slave has been selected and the transfer is underway. After writing data into the transmit FIFO, the shift logic does not begin the serial transfer until a positive edge of the `sclk_out` signal is present. The delay in waiting for this positive edge depends on the baud rate of the serial transfer. Before polling the BUSY status, you should first poll the TFE status (waiting for 1) or wait for $\text{BAUDR} * \text{ssi_clk}$ clock cycles.

19.2.3.1.3 Master SPI Serial Transfers

When the transfer mode is “transmit and receive” or “transmit only” ($\text{TMOD} = 2'b00$ or $\text{TMOD} = 2'b01$, respectively), transfers are terminated by the shift control logic when the transmit FIFO is empty. For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level ([TXFTLR](#)) can be used to early interrupt (`ssi_txe_intr`) the processor indicating that the transmit FIFO buffer is nearly empty.

When a DMA is used for APB accesses, the transmit data level (DMATDLR) can be used to early request (`dma_tx_req`) the DMA Controller, indicating that the transmit FIFO is nearly empty. The FIFO can then be refilled with data to continue the serial transfer. The user may also write a block of data (at least two FIFO entries) into the transmit FIFO before enabling a serial slave. This ensures that serial transmission does not begin until the number of data-frames that make up the continuous transfer are present in the transmit FIFO.

When the transfer mode is “receive only” ($\text{TMOD} = 2'b10$), a serial transfer is started by writing one “dummy” data word into the transmit FIFO when a serial slave is selected. The `tx_d` output from the SPI is held at a constant logic level for the duration of the serial transfer. The transmit FIFO is popped only once at the beginning and may remain empty for the duration of the serial transfer. The end of the serial transfer is controlled by the “number of data frames” (NDF) field in control register 1 ([CTRLR1](#)).

If, for example, you want to receive 24 data frames from a serial-slave peripheral, you should program the NDF field with the value 23; the receive logic terminates the serial transfer when the number of frames received is equal to the NDF value + 1. This transfer mode increases the bandwidth of the APB bus as the transmit FIFO never needs to be serviced during the transfer. The receive FIFO buffer should be read each time the receive FIFO generates a FIFO full interrupt request to prevent an overflow.

The receive FIFO threshold level (RXFTLR) can be used to give early indication that the receive FIFO is nearly full. When a DMA is used for APB accesses, the receive data level (DMARDLR) can be used to early request (`dma_rx_req`) the DMA Controller, indicating that the receive FIFO is nearly full.

A typical software flow for completing an SPI serial transfer from the SPI serial master is outlined as follows:

- (1) If the SPI is enabled, disable it by writing 0 to the SSI Enable register ([SSIENR](#)).
- (2) Set up the SPI control registers for the transfer; these registers can be set in any order.
 - Write Control Register 0 ([CTRLR0](#)). For SPI transfers, the serial clock polarity and serial clock phase parameters must be set identical to target slave device.
 - If the transfer mode is receive only, write Control Register 1 ([CTRLR1](#)) with the number of frames in the transfer minus 1; for example, if you want to receive four data frames, write this register with 3.
 - Write the Baud Rate Select Register ([BAUDR](#)) to set the baud rate for the transfer.
 - Write the Transmit and Receive FIFO Threshold Level registers ([TXFTLR](#) and [RXFTLR](#), respectively) to set FIFO threshold levels.
 - Write the IMR register to set up interrupt masks.

- The Slave Enable Register ([SER](#)) can be written here to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the Data Register ([DR](#)), the transfer does not begin until a slave is enabled.
- (3) Enable the SPI by writing 1 to the SSIENR register.
- (4) Write data for transmission to the target slave into the transmit FIFO (write DR).
If no slaves were enabled in the SER register at this point, enable it now to begin the transfer.
- (5) Poll the BUSY status to wait for completion of the transfer. The BUSY status cannot be polled immediately; for more information, see the note on page 40.
If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
- (6) The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is receive only (TMOD = 2'b10), the transfer is stopped by the shift control logic when the specified number of frames have been received. When the transfer is done, the BUSY status is reset to 0.
- (7) If the transfer mode is not transmit only (TMOD != 01), read the receive FIFO until it is empty.
- (8) Disable the SPI by writing 0 to SSIENR.

19.2.3.2 Serial-Slave Mode

SPI slave enables serial communication with master peripheral devices. When the SPI is configured as a slave device, all serial transfers are initiated and controlled by the serial bus master.

Ameba-D SPI slave can only operates in transmit and receive mode. That is, TMOD field in [CTRLR0](#) register is invalid for SPI slave. If you do not want this device to respond with data, slave output can be disabled through SLV_OE bit in [CTRLR0](#).

When the SPI serial slave is selected during configuration, it enables its txd data onto the serial bus. All data transfers to and from the serial slave are regulated on the serial clock line (sclk_in), driven from the serial-master device. Data are propagated from the serial slave on one edge of the serial clock line and sampled on the opposite edge.

When the SPI serial slave is not selected, it must not interfere with data transfers between the serial-master and other serial-slave devices. When the serial slave is not selected, its txd output is buffered, resulting in a high impedance drive onto the serial master rxd line.

The serial clock that regulates the data transfer is generated by the serial-master device and input to the SPI slave on sclk_in. The slave remains in an idle state until selected by the bus master. When not actively transmitting data, the slave must hold its txd line in a high impedance state to avoid interference with serial transfers to other slave devices. The slave continues to transfer data to and from the master device as long as it is selected.

19.2.3.2.1 Slave SPI Serial Transfers

If the SPI slave transmits data to the master, you must ensure that data exists in the transmit FIFO before a transfer is initiated by the serial-master device. If the master initiates a transfer to the SPI slave when no data exists in the transmit FIFO, an error flag (TXE) is set in the SPI status register, and the previously transmitted data frame is resent on txd. For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level register (TXFTLR) can be used to early interrupt (ssi_txe_intr) the processor, indicating that the transmit FIFO buffer is nearly empty. When a DMA Controller is used for APB accesses, the DMA transmit data level register (DMATDLR) can be used to early request (dma_tx_req) the DMA Controller, indicating that the transmit FIFO is nearly empty. The FIFO can then be refilled with data to continue the serial transfer.

The receive FIFO buffer should be read each time the receive FIFO generates a FIFO full interrupt request to prevent an overflow. The receive FIFO threshold level register (RXFTLR) can be used to give early indication that the receive FIFO is nearly full. When a DMA Controller is used for APB accesses, the DMA receive data level register (DMARDLR) can be used to early request (dma_rx_req) the DMA controller, indicating that the receive FIFO is nearly full.

A typical software flow for completing a continuous serial transfer from a serial master to the SPI slave is described as follows:

- (1) If the SPI slave is enabled, disable it by writing 0 to the SSI Enable register (SSIENR).
- (2) Set up the SPI control registers for the transfer. These registers can be set in any order.
 - Write CTRLR0 (for SPI transfers SCPH and SCPOL must be set identical to the master device).
 - Write TXFTLR and RXFTLR to set FIFO threshold levels.
 - Write the IMR register to set up interrupt masks.
- (3) Enable the SPI slave by writing 1 to SSIENR.

- (4) Write data to the data register (DR)
- (5) The SPI slave is now ready for the serial transfer. The transfer begins when the SPI slave is selected by a serial-master device.
- (6) When the transfer is underway, the BUSY status can be polled to return the transfer status. If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
- (7) The transfer ends when the serial master removes the select input to the SPI slave. When the transfer is completed, the BUSY status is reset to 0.
- (8) Disable the SPI slave by writing 0 to SSIENR.

19.2.4 DMA Controller Interface

The SPI has optional built-in DMA capability which can be selected at configuration time; it has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA.

The SPI uses two DMA channels, one for the transmit data and one for the receive data. The SPI has these DMA registers:

- [DMACR](#) – Control register to enable DMA operation.
- [DMATDLR](#) – Register to set the transmit the FIFO level at which a DMA request is made.
- [DMARDLR](#) – Register to set the receive FIFO level at which a DMA request is made.

To enable the DMA Controller interface on the SPI, you must write the DMA Control Register (DMACR). Writing a 1 into the TDMAE bit field of DMACR enables the SPI transmit handshaking interface. Writing a 1 into the RDMAE bit field of DMACR enables the SPI receive handshaking interface.

19.2.4.1 Transmit Watermark Level and Transmit FIFO Underflow

During SPI serial transfers, transmit FIFO requests are made to the DMA Controller whenever the number of entries in the transmit FIFO is less than or equal to the DMA Transmit Data Level Register (DMATDLR) value; this is known as the watermark level. The DMA Controller responds by writing a burst of data to the transmit FIFO buffer.

Data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously; that is, when the FIFO begins to empty another DMA request should be triggered. Otherwise the FIFO will run out of data (underflow). To prevent this condition, the user must set the watermark level correctly.

19.2.4.2 Receive Watermark Level and Receive FIFO Overflow

During SPI serial transfers, receive FIFO requests are made to the DMA Controller whenever the number of entries in the receive FIFO is at or above the DMA Receive Data Level Register; that is, DMARDLR+1. This is known as the watermark level. The DMA Controller responds by fetching a burst of data from the receive FIFO buffer.

Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously; that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise, the FIFO will fill with data (overflow). To prevent this condition, the user must correctly set the watermark level.

19.3 Registers

This section describes the programmable registers of the SPI.

19.3.1 Register Memory Map

Table 3-2 provides the details of the SPI memory map. All registers in the SPI are addressed at 32-bit boundaries to remain consistent with the AHB bus. Where the physical size of any register is less than 32-bit wide, the upper unused bits of the 32-bit boundary are reserved. Writing to these bits has no effect; reading from these bits returns 0.

The base address of SPI0 is 0x4007_8000, and the size is 2K; the base address of SPI1 is 0x4000_E000, and the size is 2K.

Table 19-1 Memory map of SPI

Name	Address Offset	Access	Width	Reset	Description
CTRLR0	0x00	R/W	32 bits	Configuration dependent for some bit fields	Control Register 0
CTRLR1	0x04	R/W	16 bits	0x0	Control Register 1
SSIENR	0x08	R/W	1 bits	0x0	SSI Enable Register
RSVD	0x0C	N/A	N/A	-	Reserved
SER	0x10	R/W	1 bits	0x0	Slave Enable Register
BAUDR	0x14	R/W	16 bits	0x0	Baud Rate Select
TXFTLR	0x18	R/W	6 bits	0x0	Transmit FIFO Threshold Level
RXFTLR	0x1C	R/W	6 bits	0x0	Receive FIFO Threshold Level
TXFLR	0x20	R	7 bits	0x0	Transmit FIFO Level Register
RXFLR	0x24	R	7 bits	0x0	Receive FIFO Level Register
SR	0x28	R	7 bits	0x6	Status Register
IMR	0x2C	R/W	6 bits: when SSI_IS_MASTER = 1 8 bits: when SSI_IS_MASTER = 0	0x3F/0x1F	Interrupt Mask Register
ISR	0x30	R	6 bits: when SSI_IS_MASTER = 1 8 bits: when SSI_IS_MASTER = 0	0x0	Interrupt Status Register
RISR	0x34	R	6 bits	0x0	Raw Interrupt Status Register
TXOICR	0x38	R	1 bits	0x0	Transmit FIFO Overflow Interrupt Clear Register
RXOICR	0x3C	R	1 bits	0x0	Receive FIFO Overflow Interrupt Clear Register
RXUICR	0x40	R	1 bits	0x0	Receive FIFO Underflow Interrupt Clear Register
MSTICR/FAEICR	0x44	R	1 bits	0x0	Multi-Master Interrupt Clear Register/Frame Alignment Error Interrupt Clear Register
ICR	0x48	R	1 bits	0x0	Interrupt Clear Register
DMACR	0x4C	R/W	2 bits	0x0	DMA Control Register
DMATDLR	0x50	R/W	6 bits	0x0	DMA Transmit Data Level
DMARDLR	0x54	R/W	6 bits	0x0	DMA Receive Data Level
TXUICR	0x58	R	1 bits	Not affected by reset	Transmit FIFO Underflow Interrupt Clear Register
SSRICR	0x5C	R	1 bits	See the releases table in the <i>AMBA 2 release notes</i>	SS_N Rising Edge Detect Interrupt Clear Register
DR	0x60 – 0xEC	R/W	16 bits	0x0	Data Register Note: If the Data Register (DR) is accessed from an AHB master (such as a DMA controller or a processor), the AHB transfer type may be a burst. During AHB burst transfers, the address increments after each beat of the burst. To facilitate an AHB burst, read, or write operation to the transmit or receive FIFO, the Data Register occupies thirty-six 32-bit address locations of memory map. Each of the 16-bit address locations are aliased to the DR; single accesses to the DR may use any of the 16-bit address locations.

RX_SAMPLE_DLY	0xF0	R/W	8 bits	0x0	rx Sample Delay Register
RSVD_0	0xF4	R/W	32 bits	Writes have no effect; reads return value of 0.	Reserved location for future use
RSVD_1	0xF8	R/W	32 bits	Writes have no effect; reads return value of 0.	Reserved location for future use
RSVD_2	0xFC	R/W	32 bits	Writes have no effect; reads return value of 0.	Reserved location for future use

19.3.2 Registers and Field Descriptions

The following sections contain the memory diagrams and field descriptions for the individual registers.

19.3.2.1 CTRLR0

- **Name:** Control Register 0
- **Size:** 32 bits
- **Address offset:** 0x0
- **Read/Write access:** read/write

This register controls the serial data transfer. It is impossible to write to this register when the SPI is enabled. The SPI is enabled and disabled by writing to the SSIENR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS_T	RSVD						RXBITSWAP	RXBYTESWAP	TXBITSWAP	TXBYTESWAP	RSVD				
R/W							R/W	R/W	R/W	R/W					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				SLV_OE	TMOD		SCPOL		SCPH		FRF		DFS		
				R/W	R/W		R/W		R/W		R/W		R/W		

Bit	Name	Access	Reset	Description
31	SS_T	R/W	0	When SCPH is 0 (Relevant only when the SPI is configured as a serial-master device) 0 – ss_n_out doesn't toggle between successive frames 1 – ss_n_out does toggle between successive frames
30:25	RSVD	N/A	-	Reserved
24	RXBITSWAP	R/W	0	0 – Order of receive bit doesn't swap 1 – Order of receive bit does swap
23	RXBYTESWAP	R/W	0	0 – Order of receive byte doesn't swap 1 – Order of receive byte does swap
22	TXBITSWAP	R/W	0	0 – Order of transmit bit doesn't swap 1 – Order of transmit bit does swap
21	TXBYTESWAP	R/W	0	0 – Order of transmit byte doesn't swap 1 – Order of transmit byte does swap
20:11	RSVD	N/A	-	Reserved
10	SLV_OE	R/W	0	Slave Output Enable. Relevant only when the SPI is configured as a serial-slave device. When configured as a serial master, this bit field has no functionality. This bit enables or disables the setting of the ssi_oe_n output from the SPI serial slave. When SLV_OE = 1, the ssi_oe_n output can never be active. When the ssi_oe_n output controls the tri-state buffer on the txd output from the slave, a high impedance state is always present on the slave txd output when SLV_OE = 1. This is useful when the master transmits in broadcast mode (master transmits data to all slave devices). Only one slave may respond with data on the master rxd line. This bit is enabled after reset and must be disabled by software (when broadcast mode is used), if you do not want this device to respond with data. 0 – Slave txd is enabled 1 – Slave txd is disabled

9:8	TMOD	R/W	00	<p>Transfer Mode. Relevant only when the SPI is configured as a serial-master device. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. In transmit-only mode, data received from the external device is not valid and is not stored in the receive FIFO memory; it is overwritten on the next transfer. In receive-only mode, transmitted data are not valid. After the first write to the transmit FIFO, the same word is retransmitted for the duration of the transfer. In transmit-and-receive mode, both transmit and receive data are valid. The transfer continues until the transmit FIFO is empty. Data received from the external device are stored into the receive FIFO memory, where it can be accessed by the host processor.</p> <p>00 – Transmit & Receive 01 – Transmit Only 10 – Receive Only</p>
7	SCPOL	R/W	0	<p>Serial Clock Polarity. Valid when the frame format (FRF) is set to Motorola SPI. Used to select the polarity of the inactive serial clock, which is held inactive when the SPI master is not actively transferring data on the serial bus.</p> <p>0 – Inactive state of serial clock is low 1 – Inactive state of serial clock is high</p>
6	SCPH	R/W	0	<p>Serial Clock Phase. Valid when the frame format (FRF) is set to Motorola SPI. The serial clock phase selects the relationship of the serial clock with the slave select signal.</p> <p>When SCPH = 0, data are captured on the first edge of the serial clock. When SCPH = 1, the serial clock starts toggling one cycle after the slave select line is activated, and data are captured on the second edge of the serial clock.</p> <p>0 – Serial clock toggles in middle of the first data bit 1 – Serial clock toggles at start of the first data bit</p>
5:4	FRF	R/W	00	<p>Frame Format. Selects which serial protocol transfers the data.</p> <p>00 – Motorola SPI Other – Reserved</p>
3:0	DFS	R/W	0111	<p>Data Frame Size. Selects the data frame length. When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data. For the field decode, refer to Table 19-2.</p>

Table 19-2 DFS decode

DFS value	Description
0000	Reserved – undefined operation
0001	Reserved – undefined operation
0010	Reserved – undefined operation
0011	4-bit serial data transfer
0100	5-bit serial data transfer
0101	6-bit serial data transfer
0110	7-bit serial data transfer
0111	8-bit serial data transfer
1000	9-bit serial data transfer
1001	10-bit serial data transfer
1010	11-bit serial data transfer
1011	12-bit serial data transfer
1100	13-bit serial data transfer
1101	14-bit serial data transfer
1110	15-bit serial data transfer
1111	16-bit serial data transfer

19.3.2.2 CTRLR1

- **Name:** Control Register 1
- **Size:** 16 bits
- **Address offset:** 0x04
- **Read/Write access:** read/write

This register exists only when the SPI is configured as a master device. When the SPI is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. Control register 1 controls the end of serial transfers when in receive-only mode. It is impossible to write to this register when the SPI is enabled. The SPI is enabled and disabled by writing to the SSIENR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDF															
R/W															

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	NDF	R/W	0x0	Number of Data Frames. When TMOD = 10 or TMOD = 11, this register field sets the number of data frames to be continuously received by the SPI. The SPI continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer. When the SPI is configured as a serial slave, the transfer continues for as long as the slave is selected. Therefore, this register serves no purpose and is not present when the SPI is configured as a serial slave.

19.3.2.3 SSIENR

- **Name:** SSI Enable Register
- **Size:** 1 bit
- **Address offset:** 0x08
- **Read/Write access:** read/write

This register enables and disables the SPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															SSI_EN
															R/W

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	SSI_EN	R/W	0	SPI Enable. Enables and disables all SPI operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the SPI control registers when enabled. When disabled, the ssi_sleep output is set (after delay) to inform the system that it is safe to remove the ssi_clk, thus saving power consumption in the system.

19.3.2.4 SER

- **Name:** Slave Enable Register
- **Size:** 1 bit
- **Address offset:** 0x10
- **Read/Write access:** read/write

This register is valid only when the SPI is configured as a master device. When the SPI is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. You cannot write to this register when SPI is busy.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD														SER	
														R/W	

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	SER	R/W	0	Slave Select Enable Flag. When this bit is set (1), the corresponding slave select line from the master is activated when a serial transfer begins. It should be noted that setting or clearing bits in this register have no effect on the corresponding slave select outputs until a transfer is started. Before beginning a transfer, you should enable the bit in this register that corresponds to the slave device with which the master wants to communicate. 1 – Selected 0 – Not selected

19.3.2.5 BAUDR

- **Name:** Baud Rate Select
- **Size:** 16 bits
- **Address offset:** 0x14
- **Read/Write access:** read/write

This register is valid only when the SPI is configured as a master device. When the SPI is configured as a serial slave, writing to this location has no effect; reading from this location returns 0. The register derives the frequency of the serial clock that regulates the data transfer. The 16-bit field in this register defines the ssi_clk divider value. It is impossible to write to this register when the SPI is enabled. The SPI is enabled and disabled by writing to the SSIENR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCKDV															
R/W															

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	SCKDV	R/W	0x0	SPI Clock Divider. The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (sclk_out) is disabled. The frequency of the sclk_out is derived from the following equation: $\frac{F_{sclk_out}}{SCKDV} = F_{ssi_clk}$ where SCKDV is any even value between 2 and 65534.

19.3.2.6 TXFTLR

- **Name:** Transmit FIFO Threshold Level
- **Size:** 6 bits
- **Address offset:** 0x18
- **Read/write access:** read/write

This register controls the threshold value for the transmit FIFO memory. It is impossible to write to this register when the SPI is enabled. The SPI is enabled and disabled by writing to the SSIENR register.

31	30	29	28	27	26	...	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													TFT					
													R/W					

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	-	Reserved
5:0	TFT	R/W	0x0	Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt. The FIFO depth is 64; this register is sized to the number of address bits needed to access the FIFO. If you attempt to set this value greater than or equal to the depth of the FIFO, this field is not written and retains its current value. When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered. For field decode, refer to Table 19-3.

Table 19-3 TFT decode

TFT value	Description
0000_0000	ssi_txe_intr is asserted when 0 data entry is present in transmit FIFO
0000_0001	ssi_txe_intr is asserted when 1 data entry is present in transmit FIFO
0000_0010	ssi_txe_intr is asserted when 2 data entries are present in transmit FIFO
0000_0011	ssi_txe_intr is asserted when 3 data entries are present in transmit FIFO
...	...
1111_1100	ssi_txe_intr is asserted when 252 data entries are present in transmit FIFO
1111_1101	ssi_txe_intr is asserted when 253 data entries are present in transmit FIFO
1111_1110	ssi_txe_intr is asserted when 254 data entries are present in transmit FIFO
1111_1111	ssi_txe_intr is asserted when 255 data entries are present in transmit FIFO

19.3.2.7 RXFTLR

- **Name:** Receive FIFO Threshold Level
- **Size:** 6 bits
- **Address offset:** 0x1C
- **Read/write access:** read/write

This register controls the threshold value for the receive FIFO memory. It is impossible to write to this register when the SPI is enabled. The SPI is enabled and disabled by writing to the SSIENR register.

31	30	29	28	27	26	...	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													RFT					
													R/W					

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	-	Reserved
5:0	RFT	R/W	0x0	Receive FIFO Threshold. Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt. The FIFO depth is configurable in the range 2-256. This register is sized to the number of address bits needed to access the FIFO. If you attempt to set this value greater than the depth of the FIFO, this field is not written and retains its current value. When the number of receive FIFO entries is greater than or equal to this value + 1, the receive FIFO full interrupt is triggered. For field decode, refer to Table 19-4.

Table 19-4 RFT decode

RFT value	Description
0000_0000	ssi_rxf_intr is asserted when 1 or more data entry is present in receive FIFO
0000_0001	ssi_rxf_intr is asserted when 2 or more data entries are present in receive FIFO
0000_0010	ssi_rxf_intr is asserted when 3 or more data entries are present in receive FIFO
0000_0011	ssi_rxf_intr is asserted when 4 or more data entries are present in receive FIFO
...	...

1111_1100	ssi_rxf_intr is asserted when 253 or more data entries are present in receive FIFO
1111_1101	ssi_rxf_intr is asserted when 254 or more data entries are present in receive FIFO
1111_1110	ssi_rxf_intr is asserted when 255 or more data entries are present in receive FIFO
1111_1111	ssi_rxf_intr is asserted when 256 data entries are present in receive FIFO

19.3.2.8 TXFLR

- **Name:** Transmit FIFO Level Register
- **Size:** 7 bits
- **Address offset:** 0x20
- **Read/write access:** read

This register contains the number of valid data entries in the transmit FIFO memory.

31	30	29	28	27	26	...	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													TXFL						
													R						

Bit	Name	Access	Reset	Description
31:7	RSVD	N/A	-	Reserved
6:0	TXFL	R	0x0	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

19.3.2.9 RXFLR

- **Name:** Receive FIFO Level Register
- **Size:** 7 bits
- **Address offset:** 0x24
- **Read/write access:** read

This register contains the number of valid data entries in the receive FIFO memory. This register can be read at any time.

31	30	29	28	27	26	...	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													RXFL						
													R						

Bit	Name	Access	Reset	Description
31:7	RSVD	N/A	-	Reserved
6:0	RXFL	R	0x0	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO.

19.3.2.10 SR

- **Name:** Status Register
- **Size:** 7 bits
- **Address offset:** 0x28
- **Read/write access:** read

This is a read-only register used to indicate the current transfer status, FIFO status, and any transmission/reception errors that may have occurred. The status register may be read at any time. None of the bits in this register request an interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD									DCOL	TXE	RFF	RFNE	TFE	TFNF	BUSY
									R	R	R	R	R	R	R

Bit	Name	Access	Reset	Description
-----	------	--------	-------	-------------

31:7	RSVD	N/A	-	Reserved
6	DCOL	R	0	Data Collision Error. Relevant only when the SPI is configured as a master device. This bit is set if the SPI master is actively transmitting when another master selects this device as a slave. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. 0 – No error 1 – Transmit data collision error
5	TXE	R	0	Transmission Error. Set if the transmit FIFO is empty when a transfer is started. This bit can be set only when the SPI is configured as a slave device. Data from the previous transmission is resent on the txd line. This bit is cleared when read. 0 – No error 1 – Transmission error
4	RFF	R	0	Receive FIFO Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. 0 – Receive FIFO is not full 1 – Receive FIFO is full
3	RFNE	R	0	Receive FIFO Not Empty. Set when the receive FIFO contains one or more entries and is cleared when the receive FIFO is empty. This bit can be polled by software to completely empty the receive FIFO. 0 – Receive FIFO is empty 1 – Receive FIFO is not empty
2	TFE	R	1	Transmit FIFO Empty. When the transmit FIFO is completely empty, this bit is set. When the transmit FIFO contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0 – Transmit FIFO is not empty 1 – Transmit FIFO is empty
1	TFNF	R	1	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0 – Transmit FIFO is full 1 – Transmit FIFO is not full
0	BUSY	R	0	SSI Busy Flag. When set, indicates that a serial transfer is in progress; when cleared indicates that the SPI is idle or disabled. 0 – SPI is idle or disabled 1 – SPI is actively transferring data

19.3.2.11 IMR

- **Name:** Interrupt Mask Register
- **Size:**
 - 6 bits: when SSI_IS_MASTER = 1
 - 8 bits: when SSI_IS_MASTER = 0
- **Address offset:** 0x2C
- **Read/write access:** read/write

This read/write register masks or enables all interrupts generated by the SPI.

31	30	29	...	10	9	8	
RSVD							
7	6	5	4	3	2	1	0
SSRIM	TXUIM	MSTIM/FAEIM	RXFIM	RXOIM	RXUIM	TXOIM	TXEIM
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7	SSRIM	R/W	1	SS_N Rising Edge Detect Interrupt Mask. This bit field is present only if the SPI is configured as a serial-slave device.

				0 – ssi_ssr_intr is masked 1 – ssi_ssr_intr is not masked
6	TXUIM	R/W	1	Transmit FIFO Underflow Interrupt Mask. This bit field is present only if the SPI is configured as a serial-slave device. 0 – ssi_txu_intr is masked 1 – ssi_txu_intr is not masked
5	MSTIM/FAEIM	R/W	1	When SPI is configured as serial-master, this bit field is present as Multi-Master Contention Interrupt Mask. 0 – ssi_mst_intr interrupt is masked 1 – ssi_mst_intr interrupt is not masked When SPI is configured as serial-slave, this bit field is present as Frame Alignment Interrupt Mask. 0 – ssi_fae_intr interrupt is masked 1 – ssi_fae_intr interrupt is not masked
4	RXFIM	R/W	1	Receive FIFO Full Interrupt Mask. 0 – ssi_rxf_intr interrupt is masked 1 – ssi_rxf_intr interrupt is not masked
3	RXOIM	R/W	1	Receive FIFO Overflow Interrupt Mask. 0 – ssi_rxo_intr interrupt is masked 1 – ssi_rxo_intr interrupt is not masked
2	RXUIM	R/W	1	Receive FIFO Underflow Interrupt Mask. 0 – ssi_rxu_intr interrupt is masked 1 – ssi_rxu_intr interrupt is not masked
1	TXOIM	R/W	1	Transmit FIFO Overflow Interrupt Mask. 0 – ssi_txo_intr interrupt is masked 1 – ssi_txo_intr interrupt is not masked
0	TXEIM	R/W	1	Transmit FIFO Empty Interrupt Mask. 0 – ssi_txe_intr interrupt is masked 1 – ssi_txe_intr interrupt is not masked

19.3.2.12 ISR

- **Name:** Interrupt Status Register
- **Size:**
 - 6 bits: when SSI_IS_MASTER = 1
 - 8 bits: when SSI_IS_MASTER = 0
- **Address offset:** 0x30
- **Read/write access:** read

This register reports the status of the SPI interrupts after they have been masked.

31	30	29	...	10	9	8	
RSVD							
7	6	5	4	3	2	1	0
SSRIS	TXUIS	MSTIS/FAEIS	RXFIS	RXOIS	RXUIS	TXOIS	TXEIS
R	R	R	R	R	R	R	R

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7	SSRIS	R	0	SS_N Rising Edge Detect Interrupt Status. This bit field is present only if the SPI is configured as a serial-slave device. 0 – ssi_ssr_intr interrupt is not active after masking 1 – ssi_ssr_intr interrupt is active after masking
6	TXUIS	R	0	Transmit FIFO Under Flow Interrupt Status. This bit field is present only if the SPI is configured as a serial-slave device. 0 – ssi_txu_intr interrupt is not active after masking

				1 – ssi_txu_intr interrupt is active after masking
5	MSTIS/FAEIS	R	0	When SPI is configured as serial-master, this bit field is present as Multi-Master Contention Interrupt Status. 0 – ssi_mst_intr interrupt is not active after masking 1 – ssi_mst_intr interrupt is active after masking When SPI is configured as serial-slave, this bit field is present as Frame Alignment Interrupt Status. 0 – ssi_fae_intr interrupt not active after masking 1 – ssi_fae_intr interrupt is active after masking
4	RXFIS	R	0	Receive FIFO Full Interrupt Status. 0 – ssi_rxf_intr interrupt is not active after masking 1 – ssi_rxf_intr interrupt is full after masking
3	RXOIS	R	0	Receive FIFO Overflow Interrupt Status. 0 – ssi_rxo_intr interrupt is not active after masking 1 – ssi_rxo_intr interrupt is active after masking
2	RXUIS	R	0	Receive FIFO Underflow Interrupt Status. 0 – ssi_rxu_intr interrupt is not active after masking 1 – ssi_rxu_intr interrupt is active after masking
1	TXOIS	R	0	Transmit FIFO Overflow Interrupt Status. 0 – ssi_txo_intr interrupt is not active after masking 1 – ssi_txo_intr interrupt is active after masking
0	TXEIS	R	0	Transmit FIFO Empty Interrupt Status. 0 – ssi_txe_intr interrupt is not active after masking 1 – ssi_txe_intr interrupt is active after masking

19.3.2.13 RISR

- **Name:** Raw Interrupt Status Register
- **Size:** 8 bits
- **Address offset:** 0x34
- **Read/write access:** read

This read-only register reports the status of the SPI interrupts prior to masking.

31	30	29	...		10	9	8
RSVD							
7	6	5	4	3	2	1	0
SSRIR	TXUIR	MSTIR/FAEIR	RXFIR	RXOIR	RXUIR	TXOIR	TXEIR
R	R	R	R	R	R	R	R

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7	SSRIR	R	0	SS_N Rising Edge Detect Raw Interrupt Status. This bit field is present only if the SPI is configured as a serial-slave device. 0 – ssi_ssr_intr interrupt is not active prior to masking 1 – ssi_ssr_intr interrupt is active prior to masking
6	TXUIR	R	0	Transmit FIFO Under Flow Raw Interrupt Status. This bit field is present only if the SPI is configured as a serial-slave device. 0 – ssi_txu_intr interrupt is not active prior to masking 1 – ssi_txu_intr interrupt is active prior to masking
5	MSTIR/FAEIR	R	0	When SPI is configured as serial-master, this bit field is present as Multi-Master Contention Raw Interrupt Status. 0 – ssi_mst_intr interrupt is not active prior to masking 1 – ssi_mst_intr interrupt is active prior to masking When SPI is configured as serial-slave, this bit field is present as Frame Alignment Error Raw Interrupt Status.

				0 – ssi_fae_intr interrupt not active prior to masking 1 – ssi_fae_intr interrupt is active prior to masking
4	RXFIR	R	0	Receive FIFO Full Raw Interrupt Status. 0 – ssi_rxf_intr interrupt is not active prior to masking 1 – ssi_rxf_intr interrupt is active prior to masking
3	RXOIR	R	0	Receive FIFO Overflow Raw Interrupt Status. 0 – ssi_rxo_intr interrupt is not active prior to masking 1 – ssi_rxo_intr interrupt is active prior to masking
2	RXUIR	R	0	Receive FIFO Underflow Raw Interrupt Status. 0 – ssi_rxu_intr interrupt is not active prior to masking 1 – ssi_rxu_intr interrupt is active prior to masking
1	TXOIR	R	0	Transmit FIFO Overflow Raw Interrupt Status. 0 – ssi_txo_intr interrupt is not active prior to masking 1 – ssi_txo_intr interrupt is active prior to masking
0	TXEIR	R	0	Transmit FIFO Empty Raw Interrupt Status. 0 – ssi_txe_intr interrupt is not active prior to masking 1 – ssi_txe_intr interrupt is active prior to masking

19.3.2.14 TXOICR

- **Name:** Transmit FIFO Overflow Interrupt Clear Register
- **Size:** 1 bit
- **Address offset:** 0x38
- **Read/write access:** read

31	30	29	28	27	26	25	...	7	6	5	4	3	2	1	0
RSVD															TXOICR
															R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	TXOICR	R	0	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_txo_intr interrupt; writing has no effect.

19.3.2.15 RXOICR

- **Name:** Receive FIFO Overflow Interrupt Clear Register
- **Size:** 1 bit
- **Address offset:** 0x3C
- **Read/write access:** read

31	30	29	28	27	26	25	...	7	6	5	4	3	2	1	0
RSVD															RXOICR
															R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	RXOICR	R	0	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxo_intr interrupt; writing has no effect.

19.3.2.16 RXUICR

- **Name:** Receive FIFO Underflow Interrupt Clear Register
- **Size:** 1 bit
- **Address offset:** 0x40
- **Read/write access:** read

31	30	29	28	27	26	25	...	7	6	5	4	3	2	1	0
RSVD														RXUICR	
														R	

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	RXUICR	R	0	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxu_intr interrupt; writing has no effect.

19.3.2.17 MSTICR/FAEICR

- **Name:** Multi-Master Interrupt Clear Register/Frame Alignment Error Interrupt Clear Register
- **Size:** 1 bit
- **Address offset:** 0x44
- **Read/write access:** read

When SPI is configured as serial-master, this register is present as MSTICR. When SPI is configured as serial-slave, this register is present as FAEICR.

31	30	29	28	27	26	...	6	5	4	3	2	1	0
RSVD													MSTICR/FAEICR
													R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	MSTICR/FAEICR	R	0	When SPI is configured as serial-master, this bit field is used to Clear Multi-Master Contention Interrupt. A read from this register clears the ssi_mst_intr interrupt; writing has no effect. When SPI is configured as serial-slave, this bit field is used to Clear Frame Alignment Interrupt. A read from this register clears the ssi_fae_intr interrupt; writing has no effect.

19.3.2.18 ICR

- **Name:** Interrupt Clear Register
- **Size:** 1 bit
- **Address offset:** 0x48
- **Read/write access:** read

31	30	29	28	27	26	25	...	7	6	5	4	3	2	1	0
RSVD														ICR	
														R	

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	ICR	R	0	Clear Interrupt. This register is set if any of the interrupts below are active. A read clears the ssi_txo_intr, ssi_rxu_intr, ssi_rxo_intr, and the ssi_mst_intr/ssi_fae_intr interrupts. Writing to this register has no effect.

19.3.2.19 DMACR

- **Name:** DMA Control Register
- **Size:** 2 bits
- **Address offset:** 0x4C
- **Read/write access:** read/write

This register is only valid when SPI is configured with a set of DMA Controller interface signals.

31	30	29	28	27	26	25	...	8	7	6	5	4	3	2	1	0
RSVD															TDMAE	RDMAE
															R/W	R/W

Bit	Name	Access	Reset	Description
31:2	RSVD	N/A	-	Reserved
1	TDMAE	R/W	0	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. 0 – Transmit DMA disabled 1 – Transmit DMA enabled
0	RDMAE	R/W	0	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel 0 – Receive DMA disabled 1 – Receive DMA enabled

19.3.2.20 DMATDLR

- **Name:** DMA Transmit Data Level Register
- **Size:** 6 bits
- **Address offset:** 0x50
- **Read/write access:** read/write

This register is only valid when the SPI is configured with a set of DMA interface signals.

31	30	29	28	27	26	25	...	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															DMATDL					
															R/W					

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	-	Reserved
5:0	DMATDL	R/W	0x0	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1. Refer to Table 19-5 for the field decode.

Table 19-5 DMATDL decode value

DMATDL value	Description
0000_0000	dma_tx_req is asserted when 0 data entry is present in transmit FIFO
0000_0001	dma_tx_req is asserted when 1 data entry is present in transmit FIFO
0000_0010	dma_tx_req is asserted when 2 data entries are present in transmit FIFO
0000_0011	dma_tx_req is asserted when 3 data entries are present in transmit FIFO
...	...
1111_1100	dma_tx_req is asserted when 252 data entries are present in transmit FIFO
1111_1101	dma_tx_req is asserted when 253 data entries are present in transmit FIFO
1111_1110	dma_tx_req is asserted when 254 data entries are present in transmit FIFO
1111_1111	dma_tx_req is asserted when 255 data entries are present in transmit FIFO

19.3.2.21 DMARDLR

- **Name:** DMA Receive Data Level Register
- **Size:** 6 bits
- **Address offset:** 0x54
- **Read/write access:** read/write

This register is only valid when SPI is configured with a set of DMA interface signals.

31	30	29	28	27	26	25	...	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															DMARDL					
															R/W					

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	-	Reserved
5:0	DMARDL	R/W	0x0	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE=1. Refer to Table 19-6 for the field decode.

Table 19-6 DMARDL decode value

DMARDL value	Description
0000_0000	dma_rx_req is asserted when 1 or more data entry is present in transmit FIFO
0000_0001	dma_rx_req is asserted when 2 or more data entries are present in transmit FIFO
0000_0010	dma_rx_req is asserted when 3 or more data entries are present in transmit FIFO
0000_0011	dma_rx_req is asserted when 4 or more data entries are present in transmit FIFO
...	...
1111_1100	dma_rx_req is asserted when 253 or more data entries are present in transmit FIFO
1111_1101	dma_rx_req is asserted when 254 or more data entries are present in transmit FIFO
1111_1110	dma_rx_req is asserted when 255 or more data entries are present in transmit FIFO
1111_1111	dma_rx_req is asserted when 256 data entries are present in transmit FIFO

19.3.2.22 TXUICR

- **Name:** Transmit FIFO Underflow Interrupt Clear Register
- **Size:** 1 bit
- **Address offset:** 0x58
- **Read/write access:** read

This register is present only if SPI is configured as serial-slave.

31	30	29	28	27	26	25	...	7	6	5	4	3	2	1	0
RSVD															TXUICR
															R

Bit	Name	Access	Reset	Description
31:1	RSVD	N/A	-	Reserved
0	TXUICR	R	0	When SPI is configured as serial-slave, this register is used to Clear Transmit FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_txu_intr interrupt; writing has no effect.

19.3.2.23 SSRICR

- **Name:** SS_N Rising Edge Detect Interrupt Clear Register
- **Size:** 1 bit
- **Address offset:** 0x5C
- **Read/write access:** read

This register is present only if SPI is configured as serial-slave.

31	30	29	28	27	26	25	...	7	6	5	4	3	2	1	0
RSVD															SSRICR
															R

Bit	Name	Access	Reset	Description
-----	------	--------	-------	-------------

31:1	RSVD	N/A	-	Reserved
0	SSRICR	R	0	When SPI is configured as serial-slave, this register is used to Clear SS_N Rinsing Edge Detect Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_ssr_intr interrupt; writing has no effect.

19.3.2.24 DR

- **Name:** Data Register
- **Size:** 16 bits
- **Address offset:** 0x60 to 0xEC
- **Read/write access:** read/write

The SPI data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SSI_EN = 1. FIFOs are reset when SSI_EN = 0.

Note: The DR register in the SPI occupies thirty-six 32-bit address locations of the memory map to facilitate AHB burst transfers. Writing to any of these address locations has the same effect as pushing the data from the pwwdata bus into the transmit FIFO. Reading from any of these locations has the same effect as popping data from the receive FIFO onto the prdata bus. The FIFO buffers on the SPI are not addressable.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR															
R/W															

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	-	Reserved
15:0	DR	R/W	0x0	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read – Receive FIFO buffer Write – Transmit FIFO buffer

19.3.2.25 RX_SAMPLE_DLY

- **Name:** Rx Sample Delay Register
- **Size:** 8 bits
- **Address offset:** 0xFC
- **Read/write access:** read/write

This register is valid only when the SPI is configured as serial-master.

This register controls the number of ssi_clk cycles that are delayed—from the default sample time—before the actual sample of the rxd input signal occurs. It is impossible to write to this register when the SPI is enabled; the SPI is enabled and disabled by writing to the SSIENR register.

31	30	29	28	27	26	...	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD													RSD							
													R/W							

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	-	Reserved
7:0	RSD	R/W	0x0	Receive Data (rxd) Sample Delay. This register is used to delay the sample of the rxd input signal. Each value represents a single ssi_clk delay on the sample of the rxd signal. Note: If this register is programmed with a value that exceeds the depth of the internal shift registers (SSI_RX_DLY_SR_DEPTH), a zero (0) delay will be applied to the rxd sample.

19.3.2.26 RSVD_0

- **Name:** Reserved location for future use
- **Size:** 32 bits
- **Address offset:** 0xF4
- **Read/write access:** N/A

This register is reserved for future use.

Bit	Name	Access	Reset	Description
31:0	N/A	N/A	-	N/A

19.3.2.27 RSVD_1

- **Name:** Reserved location for future use
- **Size:** 32 bits
- **Address offset:** 0xF8
- **Read/write access:** N/A

This register is reserved for future use.

Bit	Name	Access	Reset	Description
31:0	N/A	N/A	-	N/A

19.3.2.28 RSVD_2

- **Name:** Reserved location for future use
- **Size:** 32 bits
- **Address offset:** 0xFC
- **Read/write access:** N/A

This register is reserved for future use.

Bit	Name	Access	Reset	Description
31:0	N/A	N/A	-	N/A

20 Liquid Crystal Display Controller (LCDC)

20.1 Overall Description

20.1.1 Introduction

The LCD-TFT (Liquid Crystal Display - Thin Film Transistor) display controller provides a parallel digital 8080/RGB (Red, Green, Blue), signals for horizontal and vertical synchronization, pixel clock and data enable as output to interface directly to a variety of LCD and TFT panels.

The LCD Controller (LCDC) is used to fetch image data from RAM and output it to LCD panel. LCDC also supports LED dot matrix display. The display consists of a dot matrix of lights or mechanical indicators arranged in a rectangular configuration such that by switching on or off selected lights, text or graphics can be displayed. A dot matrix controller converts instructions from a processor into signals which turns on or off lights in the matrix so that the required display is produced.

20.1.2 Features

- Thin Film Transistor (TFT) color display
- 8-bit MCU I8080 parallel interface
 - Resolution of 8-bit mode, (1024x1024) for still picture display
 - Resolution of 8-bit mode, (645x645) for dynamic display when refresh rate is 30F/S
- 6-bit RGB parallel interface
 - Resolution of 6-bit mode, less than (527x527) for dynamic display when refresh rate is 60F/S
- RGB565 data format for input and output
- Programmable timings for different display panels
- Programmable polarity for HSYNC, VSYNC and Data Enable
- DMA frame buffer for RGB/MCU I/F
- I/O mode for MCU I/F

20.1.3 LCD Application Scenario

Ameba-D LCDC is used for menu development.

20.1.3.1 LCM with GRAM

Fig 20-1 shows the components of LCM with GRAM. In this application scenario, frame buffer is optional.

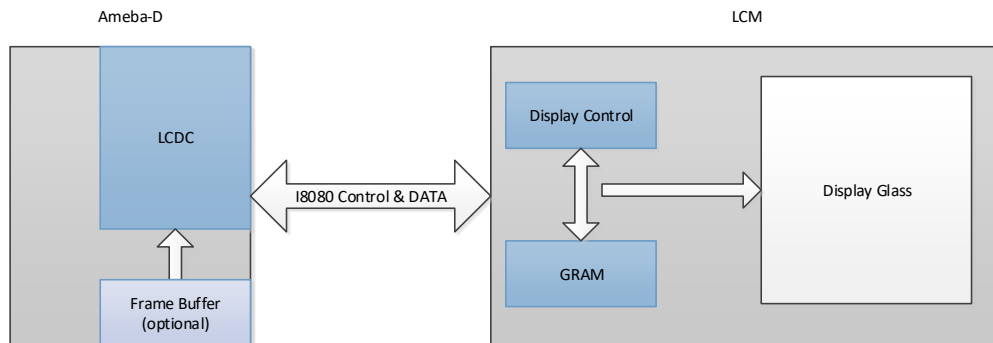


Fig 20-1 MCU I/F + LCM with GRAM

20.1.3.2 LCM without GRAM

Fig 20-2 shows the components of LCM without GRAM. In this application scenario, frame buffer is mandatory.

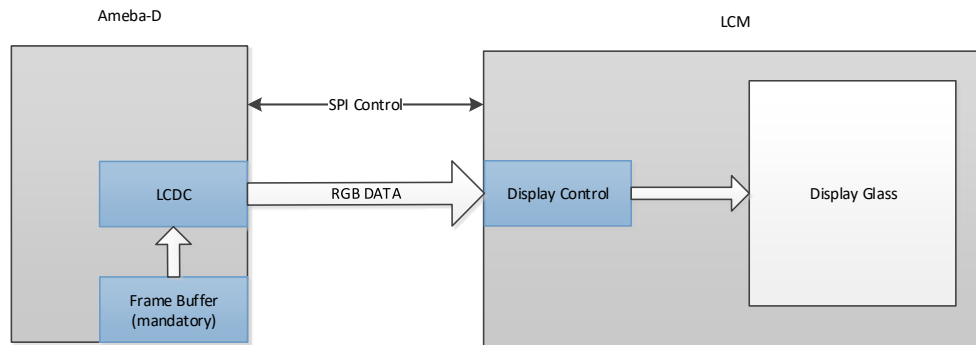


Fig 20-2 RGB I/F + LCM without GRAM

20.2 Architecture

20.2.1 Block Diagram

The block diagram of LCDC is show in Fig 20-3.

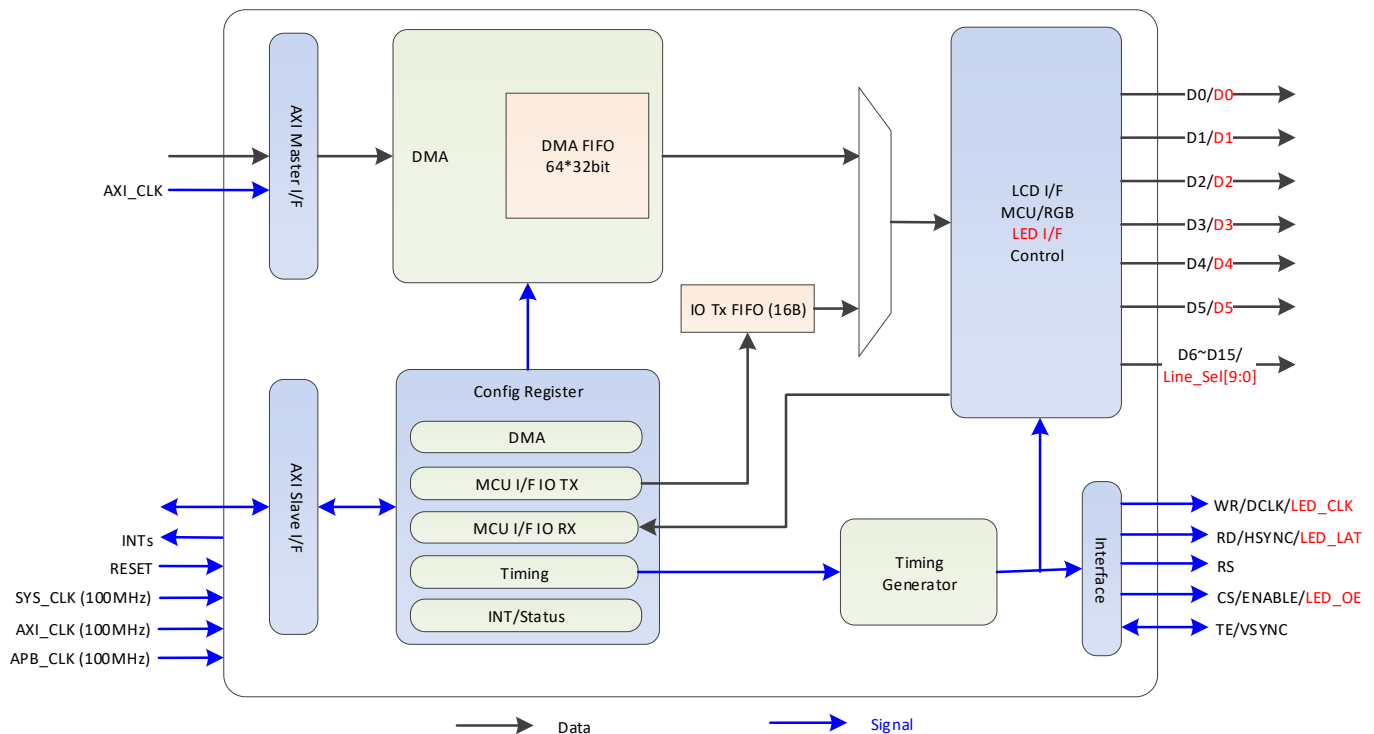


Fig 20-3 LCDC block diagram

20.2.1.1 Data Path

There are two data paths, which are shown in Fig 20-4.

- Data path 1: DMA mode

The path is Memory -> AXI Master -> LCDC -> LCD. This path is used for RGB I/F and MCU I/F DMA mode.

- Data path 2: MCU I/O mode

The path is CPU -> APB Slave -> LCDC -> LCD. This path is used for MCU I/F I/O mode, such as read/write point from LCD internal GRAM or send command to LCD.

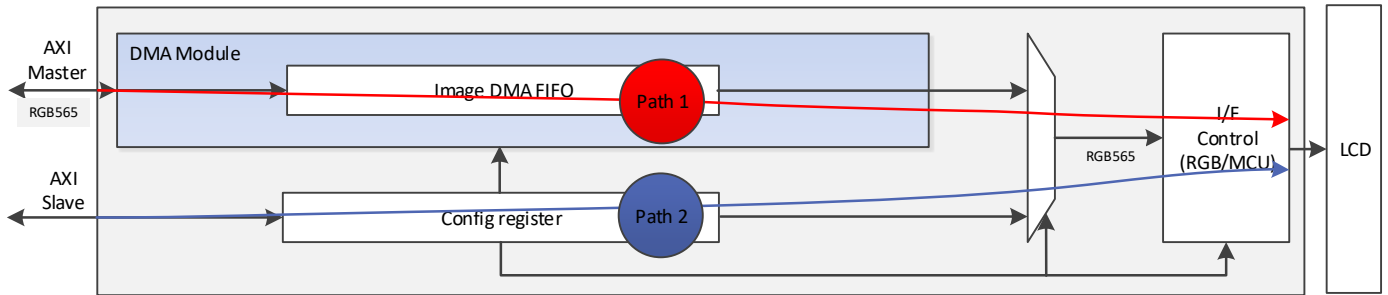


Fig 20-4 Two data paths

20.2.1.2 MCU I/O Mode

MCU I/O mode is just used for MCU I/F. The application scenario is shown in Fig 20-5.

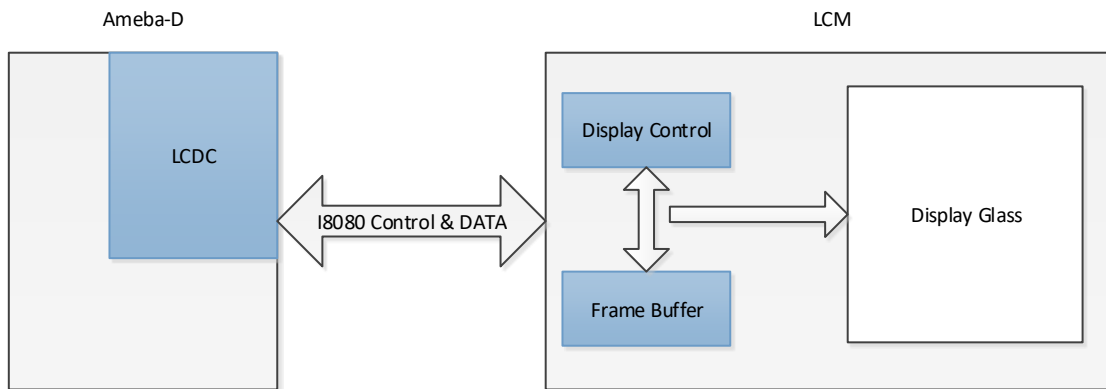


Fig 20-5 MCU I/O mode application scenario

When you use LCM (Liquid Crystal Module) with internal GRAM, you can use MCU I/F to control the image display and send image to LCD GRAM one pixel by one pixel.

If you only want to update partial image, write/read only one pixel to/from LCM. This is useful under menu development.

In this mode, you do not need to allocate a frame buffer in Ameba-D.

20.2.1.2.1 I/O Tx Mode

Every time software writes data register, the data is stored in I/O Tx FIFO temporally, and sent through hardware when MCU interface is idle.

When I/O Tx FIFO is full, I/O write action halts. When Tx FIFO is not full, write ready signal is given by hardware and then software can write another pixel data.

20.2.1.2.2 I/O Rx Mode

Every time software reads data register, one read cycle in MCU interface is triggered, so there is no Rx FIFO.

20.2.1.3 DMA Mode

The application scenario of DMA mode is shown in Fig 20-6.

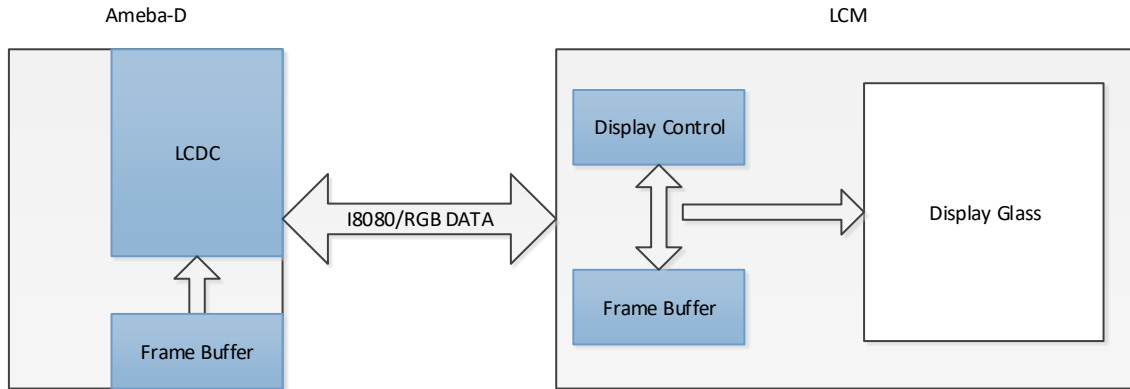


Fig 20-6 DMA mode application scenario

DMA mode is used to send the whole frame into GRAM without CPU operation. In this mode, you must allocate a frame buffer in Ameba-D.

There are two DMA modes: Auto-mode and Trigger-mode.

- Auto-mode: DMA master gets frame from frame buffer automatically based on synchronous timings.
- Trigger-mode: DMA master gets frame from frame buffer manually based on register trigger, one frame one trigger. This mode is useful when you use a LCD with internal GRAM, especially refreshing for a still image.

20.2.2 MCU Interface

The MCU interface is shown in Fig 20-7.

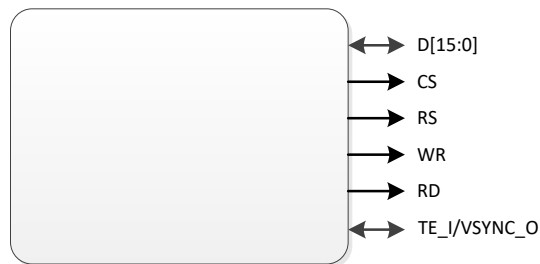


Fig 20-7 MCU interface

20.2.2.1 Write Cycle

The WR signal is driven from high to low and then pulled back to high during the write cycle. The data line is asserted at the falling edge of WR. When there is a rising edge of WR, the LCD driver reads the data line.

Write cycle includes command setting and data writing. The timing parameters of command setting and data writing are illustrated in Fig 20-8 and Fig 20-9.

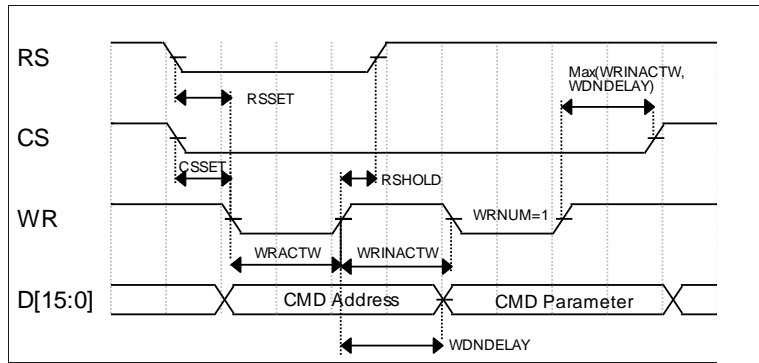


Fig 20-8 MCU I/F command setting timing parameters

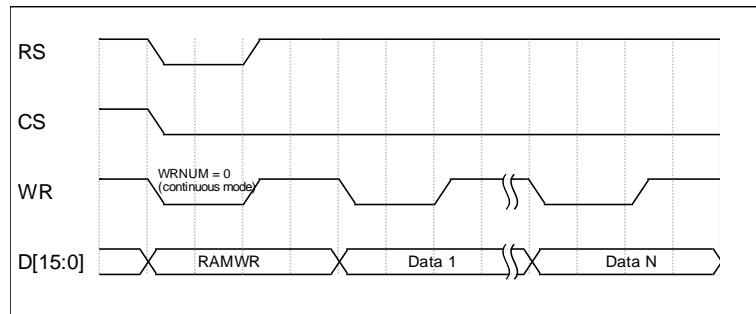


Fig 20-9 MCU I/F data writing timing parameters

20.2.2.2 Read Cycle

As Fig 20-10 shows, the RD signal is driven from high to low and then pulled back to high during the read cycle. The data line is asserted by LCD driver when there is a falling edge of RD. When there is a rising edge of RD, you can read the data line.

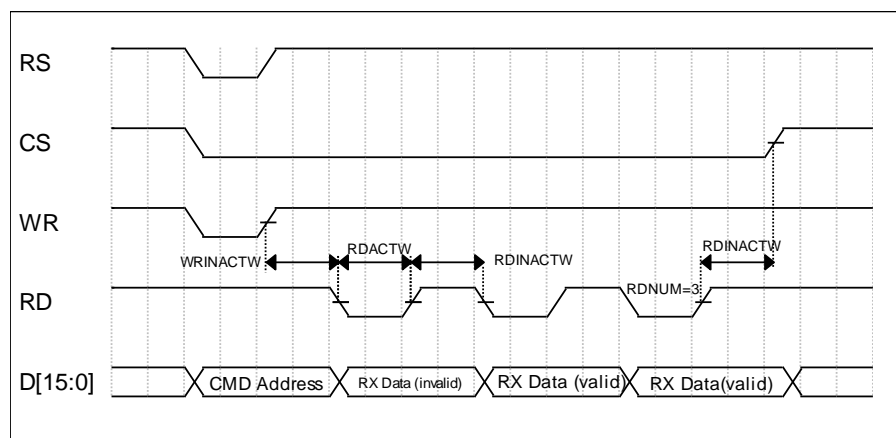


Fig 20-10 MCU I/F read command timing parameters

20.2.2.3 MCU System with VSYNC Interface

LCDC VSYNC interface starts synchronization to display the moving picture with MCU system interface according to the frame-synchronizing signal VSYNC. In this mode, LCDC controls the synchronization of frame.

The LCD display operation is synchronized with the LCDC VSYNC output and the frame rate is determined by the pulse rate of VSYNC signal. The MCU VSYNC mode timing is shown in Fig 20-11.

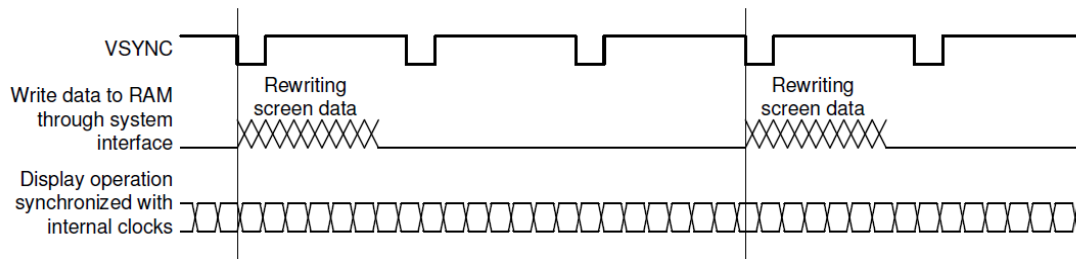


Fig 20-11 MCU VSYNC mode timing

20.2.2.4 MCU System with Tear Effect Line

Like VSYNC, Tearing Effect (TE) signal can also control the synchronization of frame. The difference is that TE outputs from LCD, but VSYNC outputs from LCDC.

TE signal can be enabled or disabled by LCD driver's related commands. LCDC output data is synchronized with LCD TE output signal. LCD TE signal is used to synchronize frame memory writing for displaying video images.

The MCU TE mode timing is shown in Fig 20-12.

- tvdh: The LCD display isn't updated from the GRAM.
- tvdl: The LCD display is updated from the GRAM (except invisible lines, see Fig 20-13).

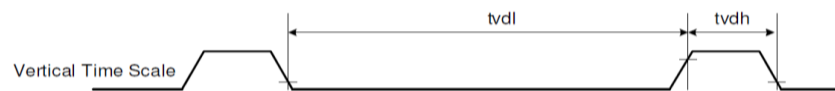


Fig 20-12 MCU TE mode timing

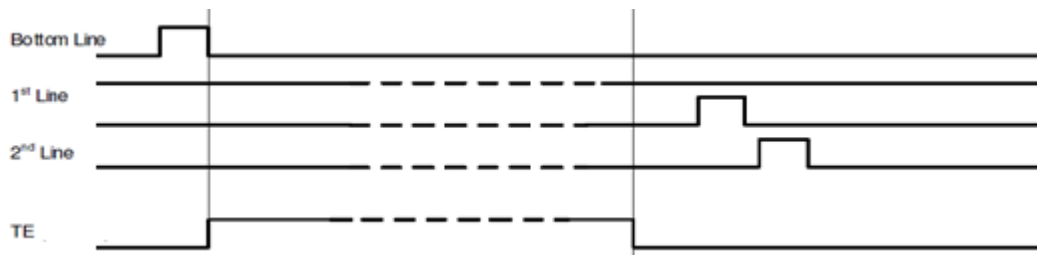


Fig 20-13 MCU TE mode frame synchronization

20.2.2.5 MCU Interface Type

Ameba-D supports different types of MCU interface, and the output pin assignments are listed in Table 20-1. The details about frame data transferring is described in the following sections.

Table 20-1 MCU interface type

I/F Type	Data Format	Support
I80 8-bit	RGB565	✓
I80 9-bit	RGB666	✗
I80 16-bit	RGB565	✓
I80 18-bit	RGB888	✗

20.2.2.6 8080 I/F 8-bit Output

The 8080 I/F 8-bit output is shown in Fig 20-14, with RGB565 bits input.

Count	0	1	2	3	4	...
RS	0	1	1	1	1	...
D7	C7	P1R4	P1G2	P2R4	P2G2	...
D6	C6	P1R3	P1G1	P2R3	P2G1	...
D5	C5	P1R2	P1G0	P2R2	P2G0	...
D4	C4	P1R1	P1B4	P2R1	P2B4	...
D3	C3	P1R0	P1B3	P2R0	P2B3	...
D2	C2	P1G5	P1B2	P2G5	P2B2	...
D1	C1	P1G4	P1B1	P2G4	P2B1	...
D0	C0	P1G3	P1B0	P2G3	P2B0	...

P1R4 : Pixel1/Red_bit4

Fig 20-14 8080 I/F 8-bit output

20.2.2.7 8080 I/F 16-bit Output

The 8080 I/F 16-bit output is shown in Fig 20-15, also with RGB565 bits input.

Count	0	1	2	...
RS	0	1	1	...
D15		P1R4	P2R4	...
D14		P1R3	P2R3	...
D13		P1R2	P2R2	...
D12		P1R1	P2R1	...
D11		P1R0	P2R0	...
D10		P1G5	P2G5	...
D9		P1G4	P2G4	...
D8		P1G3	P2G3	...
D7	C7	P1G2	P2G2	...
D6	C6	P1G1	P2G1	...
D5	C5	P1G0	P2G0	...
D4	C4	P1B4	P2B4	...
D3	C3	P1B3	P2B3	...
D2	C2	P1B2	P2B2	...
D1	C1	P1B1	P2B1	...
D0	C0	P1B0	P2B0	...

P1R4 : Pixel1/Red_bit4

Fig 20-15 8080 I/F 16-bit output

20.2.3 RGB Interface

The RGB interface is shown in Fig 20-16.



Fig 20-16 RGB interface

20.2.3.1 RGB Timing

The RGB timing is shown in Fig 20-17.

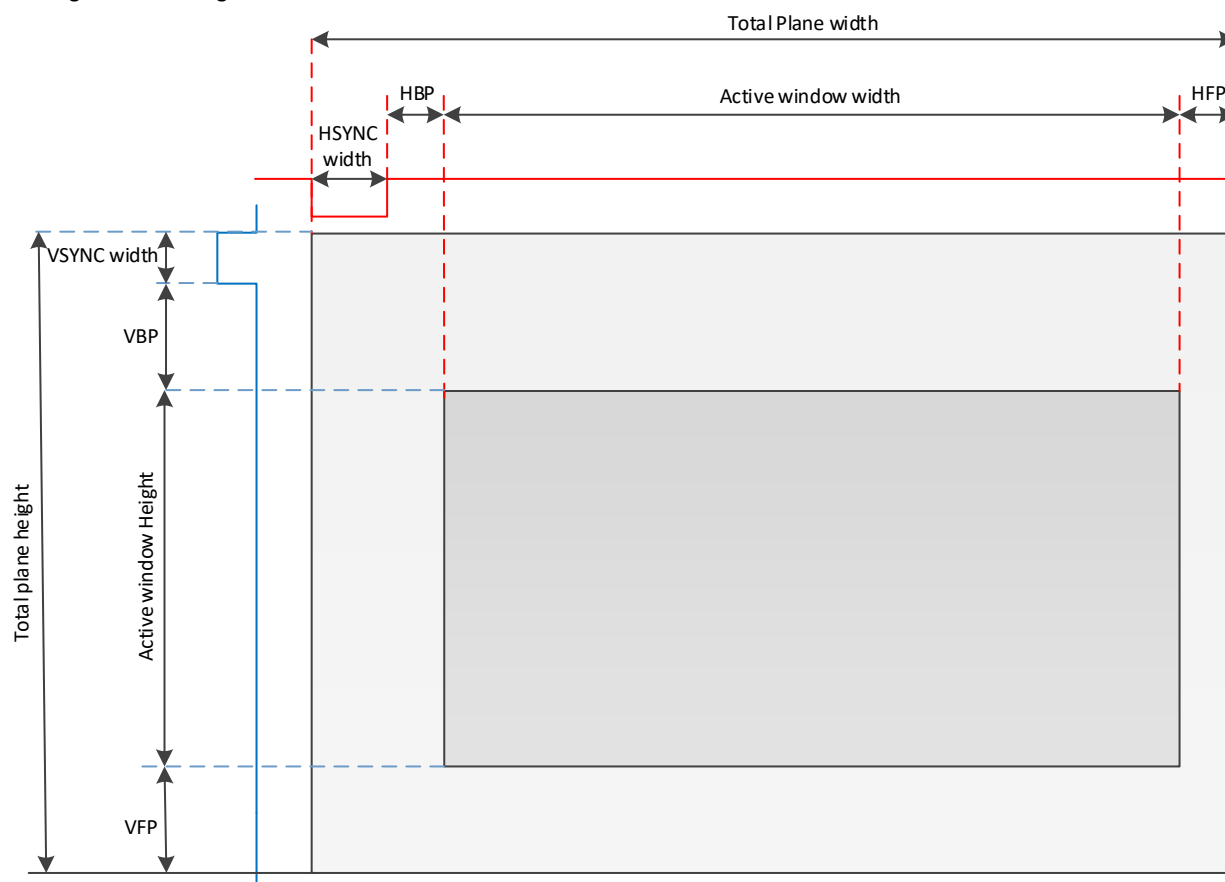


Fig 20-17 RGB timing

20.2.3.2 DE Mode

Driver determined valid data by ENABLE signal. The RGB DE mode timing is shown in Fig 20-18.

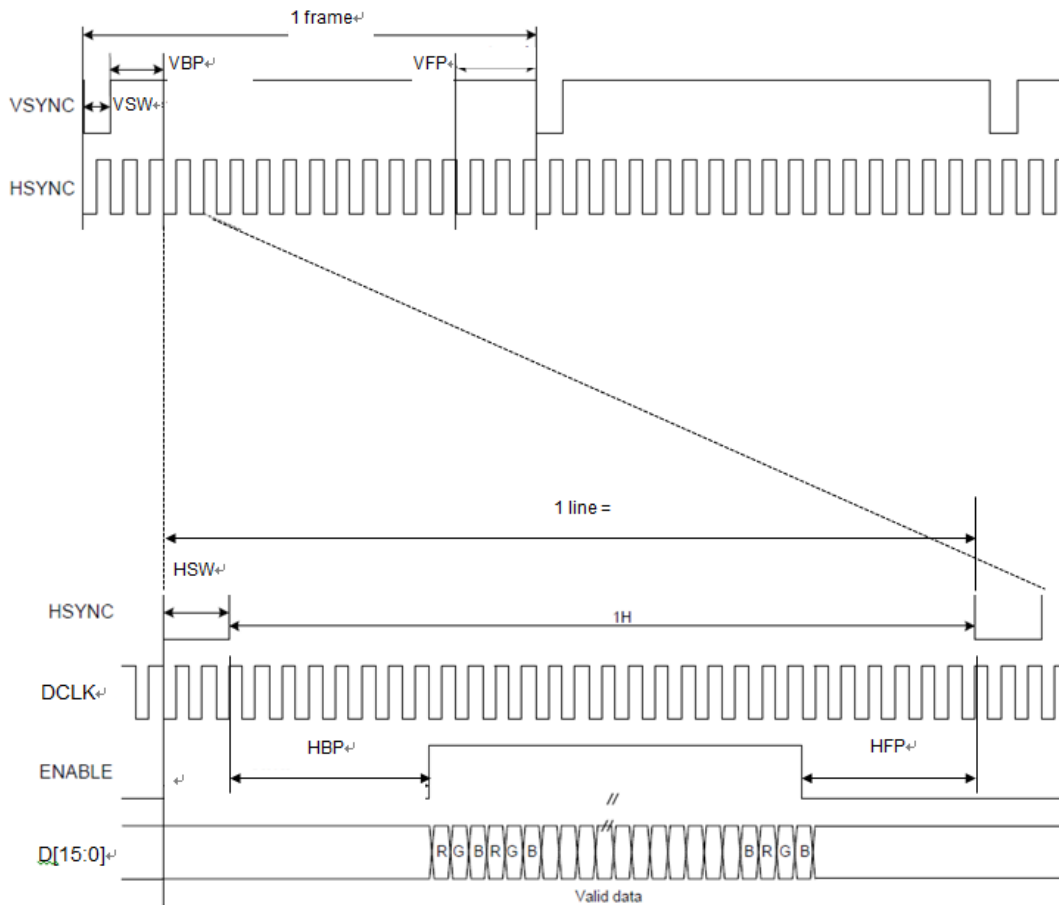


Fig 20-18 RGB DE mode timing

20.2.3.3 HV Mode

In this mode, timing is similar with DE mode but driver determined valid data by HSW/HBP/HFP settings of HSYNC signal and VSW/VBP/VFP settings of VSYNC. This mode does not need ENABLE signal.

20.2.3.4 RGB I/F 6-bit Output

The RGB I/F 6-bit output is shown in Fig 20-19, with RGB565 bits output.

Count	0	1	2	3	4	...
D5	P1R4	P1G5	P1B4	P2R4	P2G5	...
D4	P1R3	P1G4	P1B3	P2R3	P2G4	...
D3	P1R2	P1G3	P1B2	P2R2	P2G3	...
D2	P1R1	P1G2	P1B1	P2R1	P2G2	...
D1	P1R0	P1G1	P1B0	P2R0	P2G1	...
D0		P1G0			P2G0	...

P1R4 : Pixel1/Red_bit4

Fig 20-19 RGB I/F 6-bit output

20.2.3.5 RGB I/F 16-bit Output

The RGB I/F 16-bit output is shown in Fig 20-20, also with RGB565 bits output.

Count	0	1	...
D15	P1R4	P2R4	...
D14	P1R3	P2R3	...
D13	P1R2	P2R2	...
D12	P1R1	P2R1	...
D11	P1R0	P2R0	...
D10	P1G5	P2G5	...
D9	P1G4	P2G4	...
D8	P1G3	P2G3	...
D7	P1G2	P2G2	...
D6	P1G1	P2G1	...
D5	P1G0	P2G0	...
D4	P1B4	P2B4	...
D3	P1B3	P2B3	...
D2	P1B2	P2B2	...
D1	P1B1	P2B1	...
D0	P1B0	P2B0	...
P1R4 : Pixel1/Red_bit4			

Fig 20-20 RGB I/F 16-bit output

20.2.4 LED Control

LCDC also supports LED dot matrix display. The display consists of a dot matrix of lights or mechanical indicators arranged in a rectangular configuration such that by switching on or off selected lights, text or graphics can be displayed. A dot matrix controller converts instructions from a processor into signals which turns on or off lights in the matrix so that the required display is produced.

20.2.4.1 LED Module

The interface of LED module is shown in Fig 20-21.

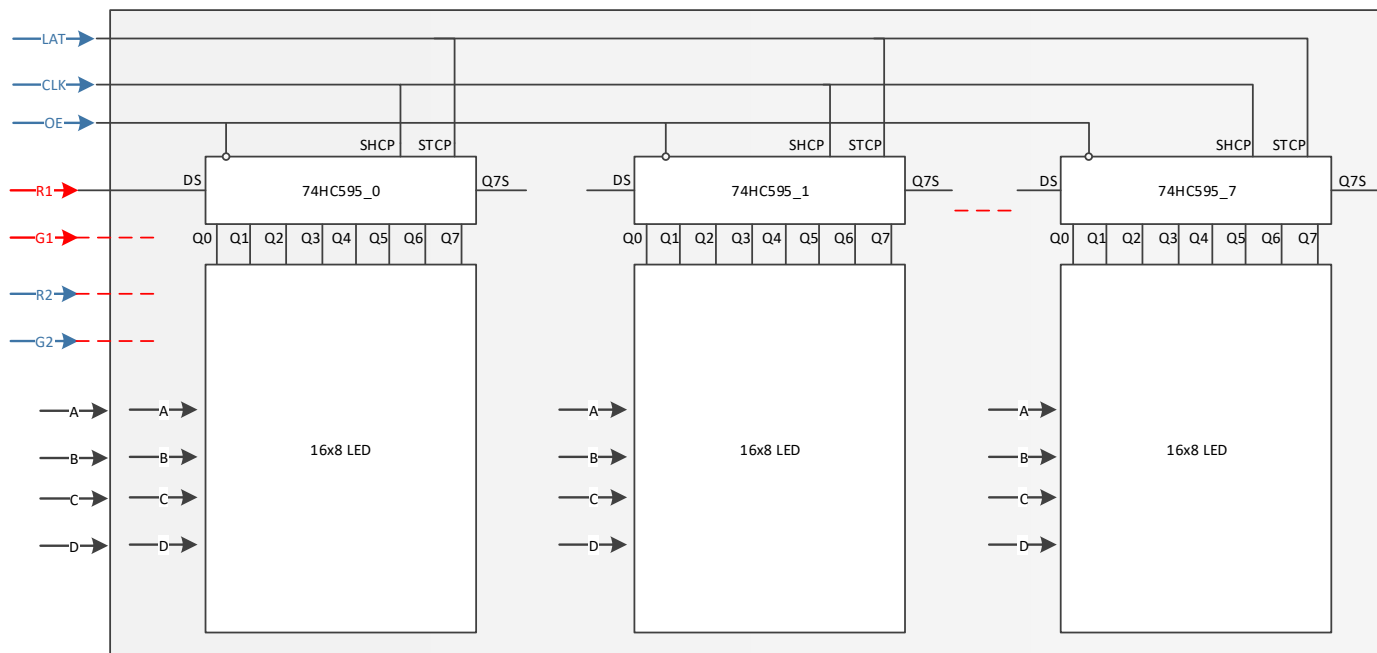


Fig 20-21 LED interface

The LED control timing is shown in Fig 20-22.

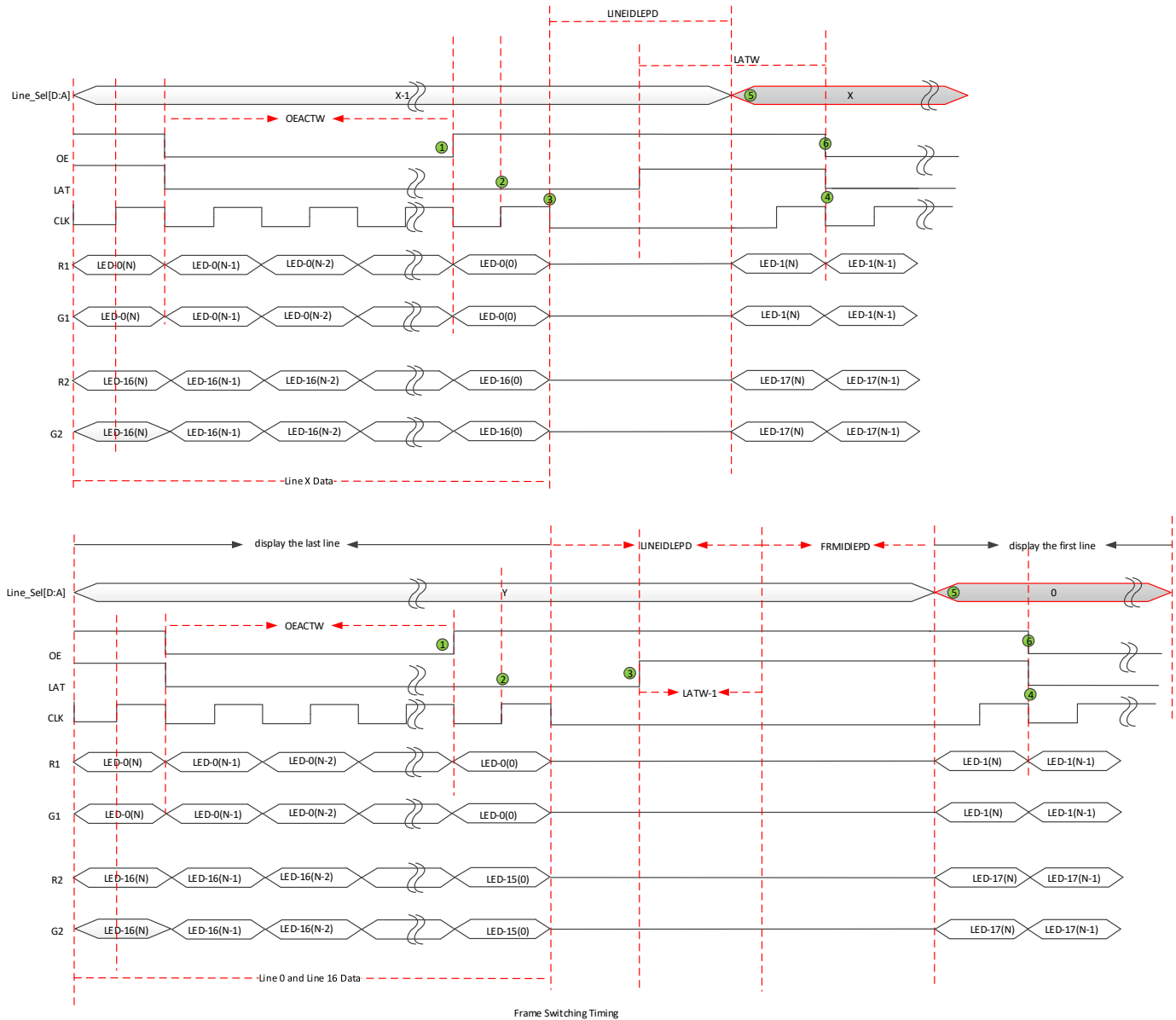


Fig 20-22 LED control timing

20.2.4.2 LED Control Diagram

The LED control diagram is shown in Fig 20-23.

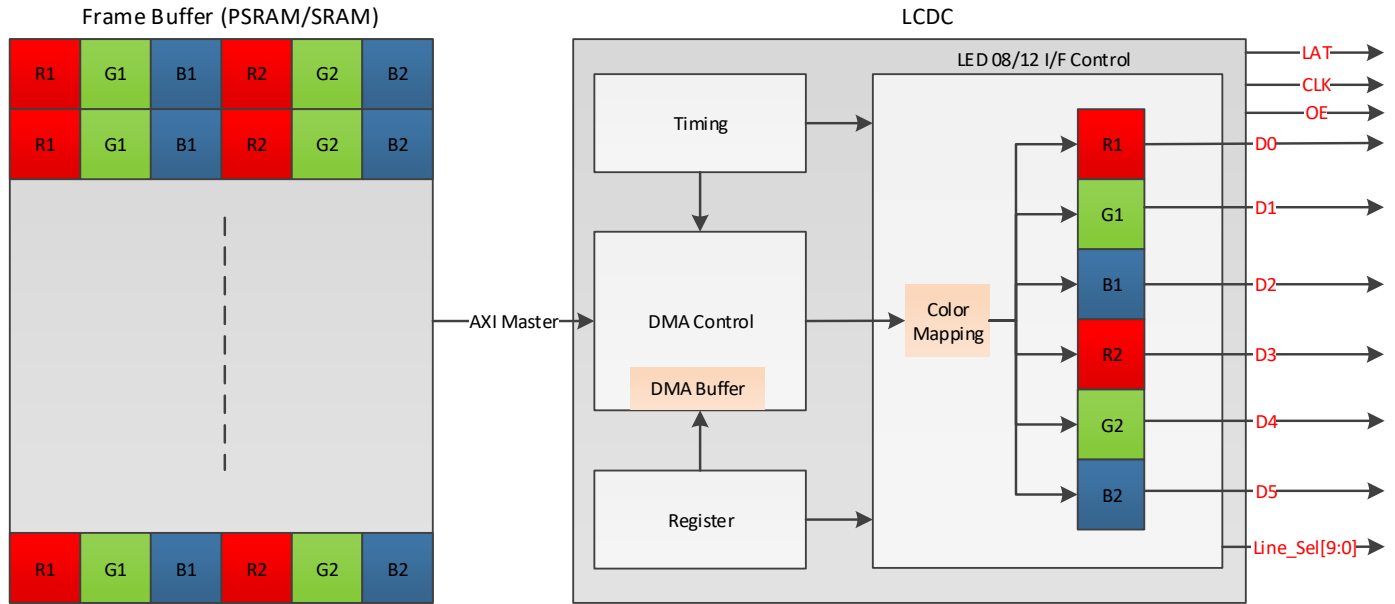


Fig 20-23 LED control diagram

20.2.4.3 LED Color Mapping

20.2.4.3.1 Single Color and Single Channel

The single color and single channel of LED color mapping is shown in Fig 20-24.

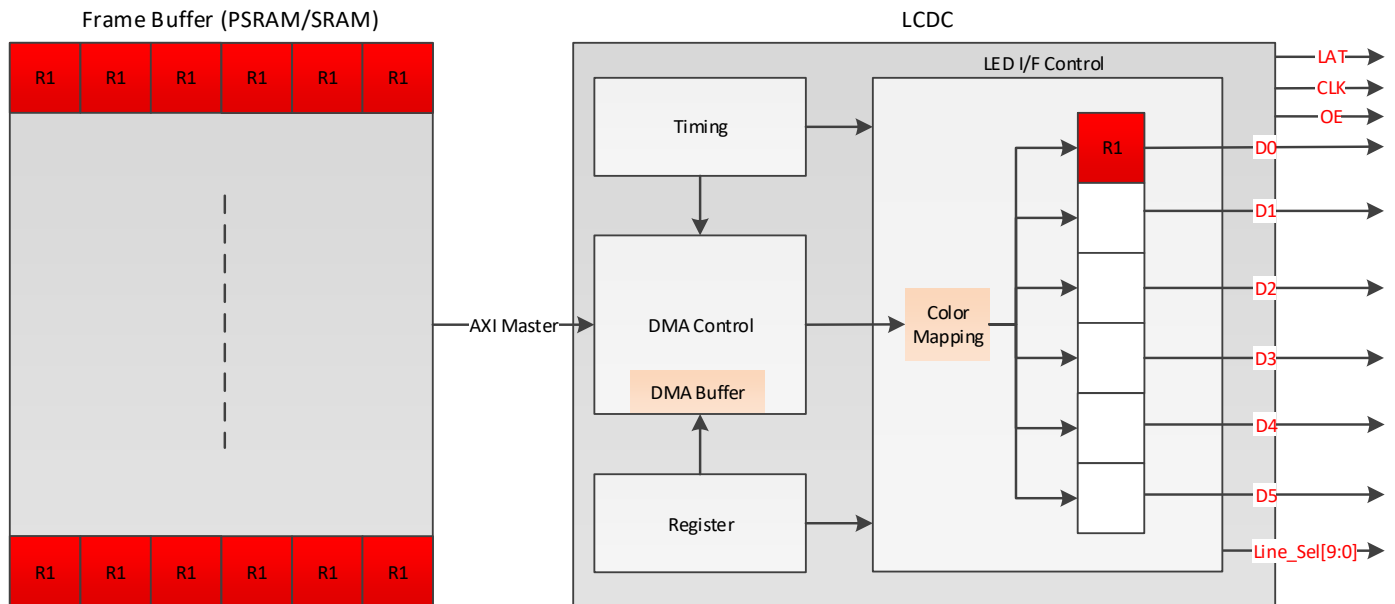


Fig 20-24 LED color mapping: single color and single channel

20.2.4.3.2 Single Color and Two Channels

The single color and two channels of LED color mapping is shown in Fig 20-25.

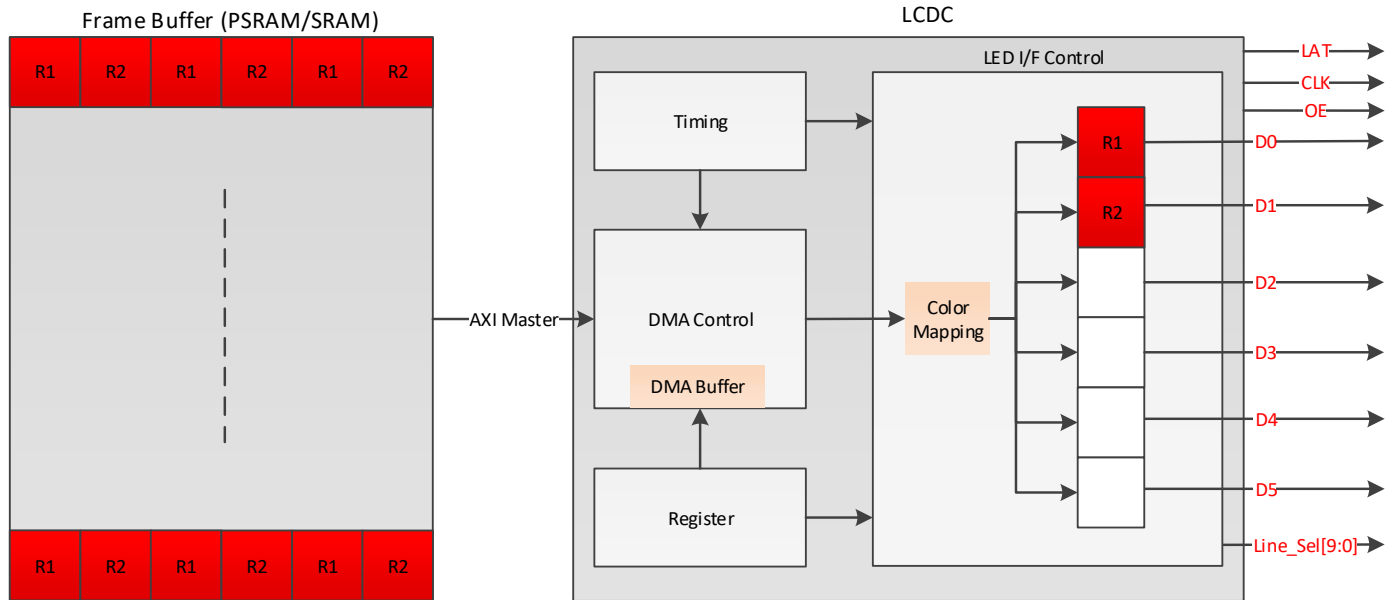


Fig 20-25 LED color capping: single color and two channels

20.2.4.3.3 Two Colors and Single Channel

The two colors and single channel of LED color mapping is shown in Fig 20-26.

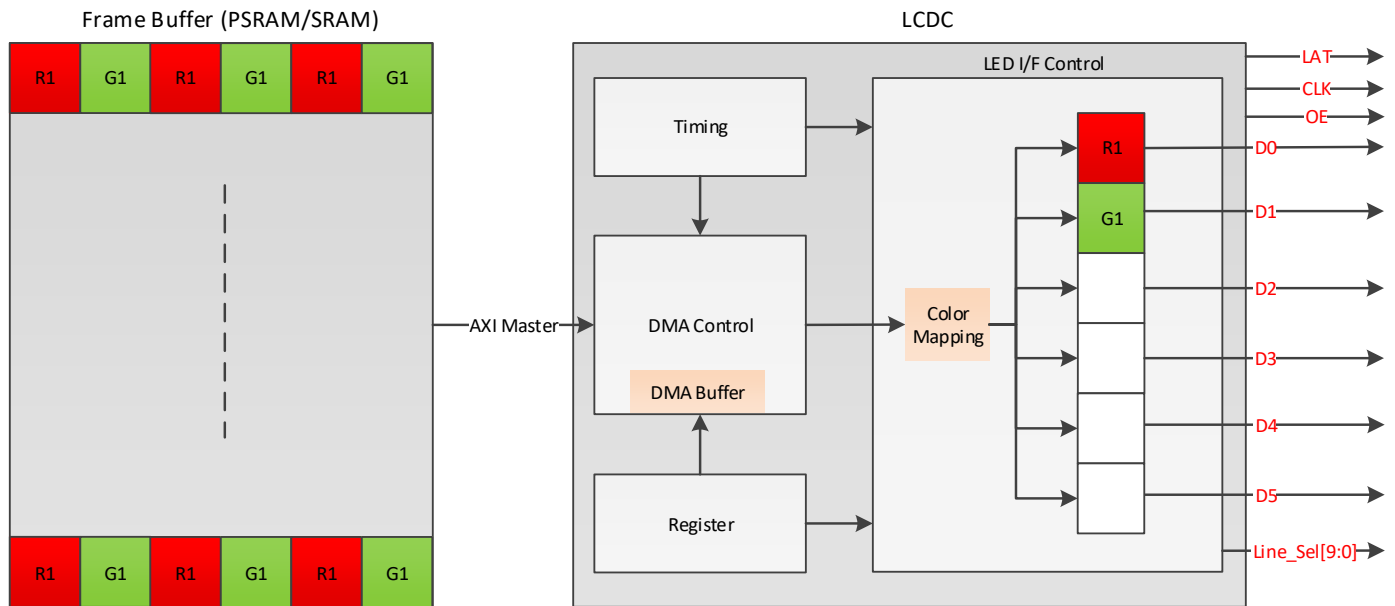


Fig 20-26 LED color mapping: two colors and single channel

20.2.4.3.4 Two Colors and Two Channels

The two colors and two channels of LED color mapping is shown in Fig 20-27.

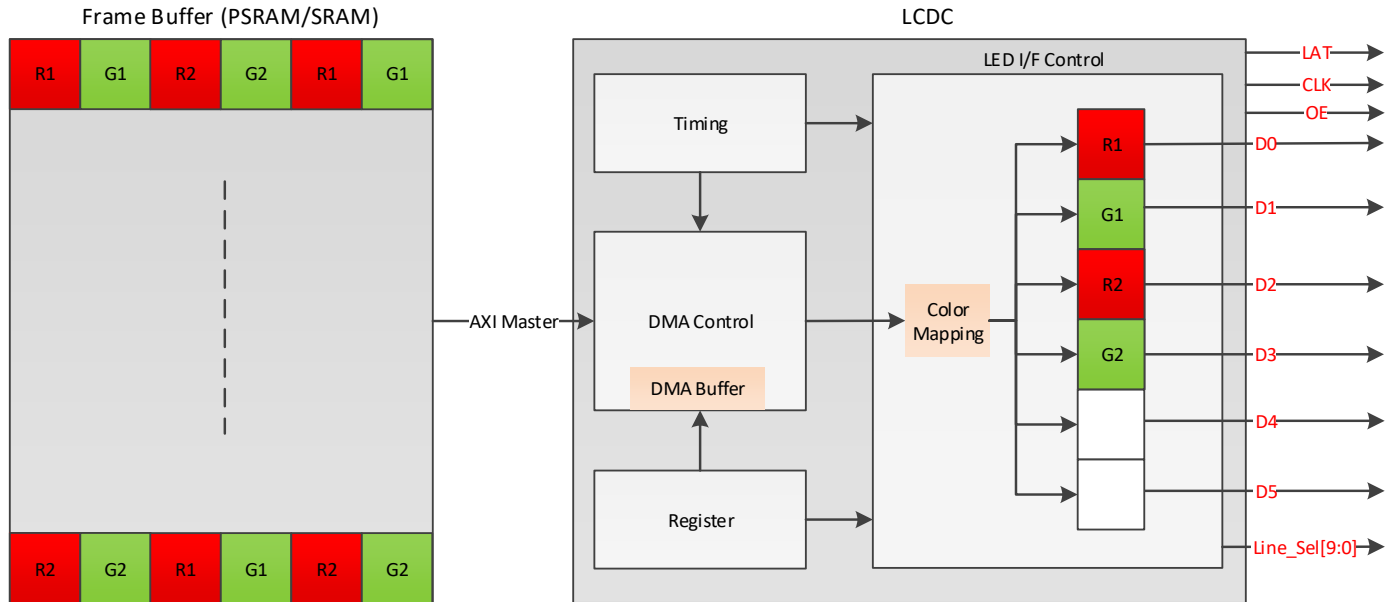


Fig 20-27 LED color mapping: two colors and two channels

20.2.4.3.5 Three Colors and Single Channel

The three colors and single channel of LED color mapping is shown in Fig 20-28.

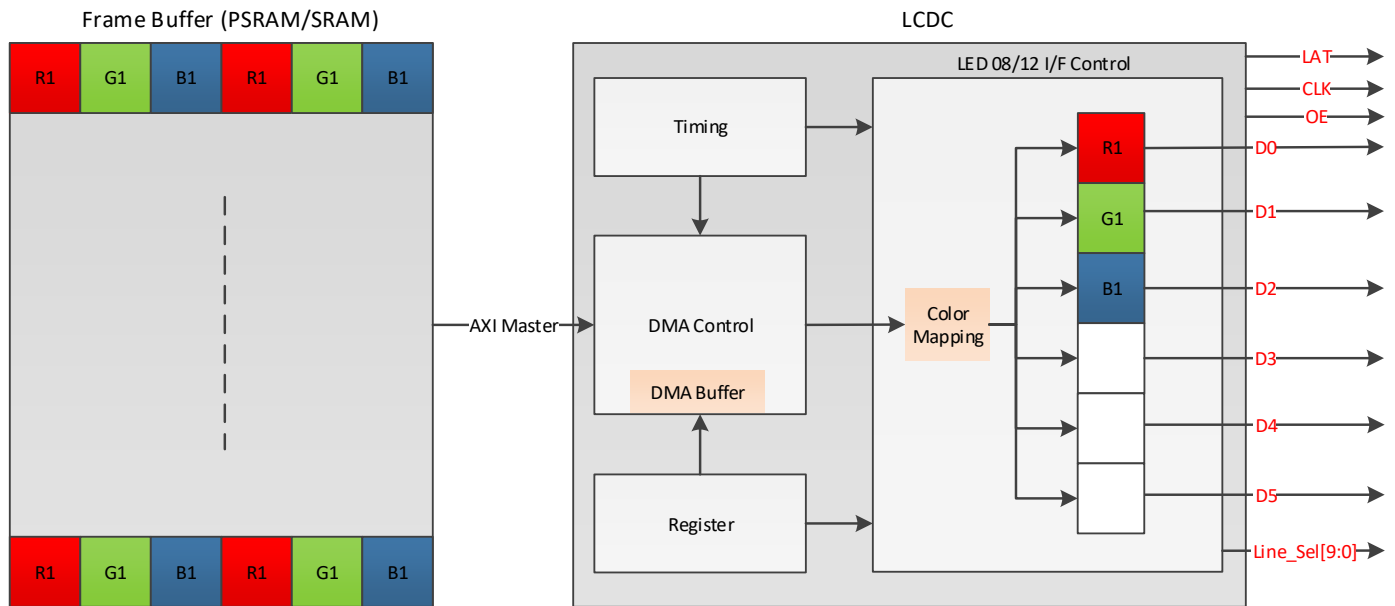


Fig 20-28 LED color mapping: three colors and single channel

20.2.4.3.6 Three Colors and Two Channels

The three colors and two channels of LED color mapping is shown in Fig 20-29.

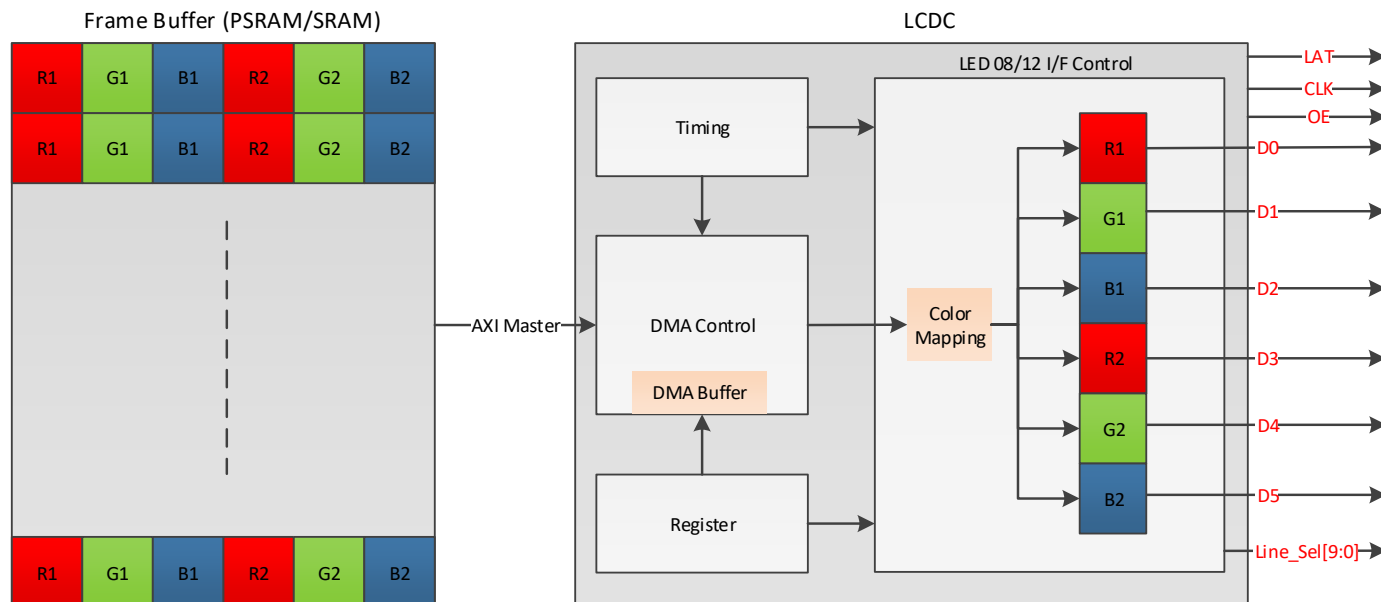


Fig 20-29 LED color mapping: three colors and two channels

20.2.5 Pinmux

The pinmux of LCDC is listed in Table 20-2.

Table 20-2 LCDC pinmux

MCU System		RGB Interface		LED Interface	
D[15:0]	I/O	D[15:0]	I/O	D[5:0]/Line_Sel[9:0]	I/O
WR	I	DCLK	O	DCLK	O
RD	I	HSYNC	O	LAT	O
RS	I				
CS	I	ENABLE	I	OE	O
TE/VSYNC	O/I	VSYNC	I		

20.2.6 Supported Resolution

20.2.6.1 MCU I/F

Parameters for maximum image size calculation:

- Max. dot clock: $\text{MAX_DOT_CLK} = \text{system_clock}/4 = 100\text{MHz}/4 = 25\text{MHz}$
- Refresh frequency: $F = 30 \text{ (F/S)}$

For 16-bit I/F mode:

$$\text{max image size} = \text{MAX DOT CLK/F} = 912 * 912$$

For 8-bit I/F mode:

$$\text{max image size} = \text{MAX DOT CLK}/(F*2) = 645*645$$

20.2.6.2 RGB I/F

Parameters for maximum image size calculation:

- Max. dot clock: $\text{MAX_DOT_CLK} = \text{system_clock}/2 = 100\text{MHz}/2 = 50\text{MHz}$
- Refresh frequency: $F = 60 \text{ (F/S)}$

For 16-bit I/F mode:

$$\text{max_image_size} = \text{MAX_DOT_CLK} / F - (\text{width} + \text{HBP} + \text{HFP}) * (\text{VBP} + \text{VBF}) - \text{height} * (\text{HBP} + \text{HFP}) < 912 * 912$$

For 6-bit I/F mode:

$$\text{max_image_size} = \text{MAX_DOT_CLK} / (3 * F) - (\text{width} + \text{HBP} + \text{HFP}) * (\text{VBP} + \text{VBF}) - \text{height} * (\text{HBP} + \text{HFP}) < 527 * 527$$

20.2.6.3 LED I/F

Parameters for maximum image size calculation:

- Max. dot clock: $\text{MAX_DOT_CLK} = \text{system_clock} / 2 = 100\text{MHz} / 2 = 50\text{MHz}$.
- Refresh frequency: $F = 100 \text{ (Hz)}$

One channel mode:

$$\text{max_image_size} = \text{MAX_DOT_CLK} / F = 707 * 707$$

Two channels mode:

$$\text{max_image_size} = (\text{MAX_DOT_CLK} * 2) / F = 1000 * 1000$$

20.2.6.4 Supported Resolution List

The maximum supported resolutions for all kinds of mode of LCDC are described in Fig 20-30.

Fig 20-30 Max. supported resolutions for LCDC

I/F	Mode	Max. Supported Resolution	Common Supported Resolution	Refresh Frequency	Note
MCU	8-bit mode	645*645	120*240, 240*320, 320*480, etc.	30Hz	<ul style="list-style-type: none"> ● The dot clock is fixed to 25MHz. ● For still picture display, the max. supported resolution can reach to 1024*1024. ● For dynamic display, the max. supported resolution is listed.
	16-bit mode	912*912	120*240, 240*320, 320*480, 640*480, etc.	30Hz	
RGB	6-bit mode	< 527*527	120*240, 240*320, etc.	60Hz	<ul style="list-style-type: none"> ● The max. dot clock is 50MHz. ● The max. supported resolution depends on the parameters (VHP, VBP, HBP, HFP) of LCD device.
	16-bit mode	< 912*912	120*240, 240*320, 320*480, 640*480, 800*480, etc.	60Hz	
LED	One channel mode	707*707	32*1024, 64*1024, 32*2048, etc.	60Hz	<ul style="list-style-type: none"> ● The max. dot clock is 50MHz. ● The max. supported resolution depends on the parameters (VHP, VBP, HBP, HFP) of LCD device.
	Two channels mode	1000*1000	32*1024, 64*1024, 32*2048, 64*2048, 128*2048, etc.	60Hz	

20.3 Registers

Table 20-3 provides the details of LCDC registers.

Table 20-3 Memory map of LCDC

Name	Address offset	Access	Description
Global Control Registers			
LCDC_CTRL	0x0000	R/W	Enable LCDC or disable LCDC (after refreshing frame done) LCD Interface mode
LCDC_PLANE_SIZE	0x0004	R/W	LCD work plane size
LCDC_UNDFLOW_CFG	0x0008	R/W	Define the DMA FIFO underflow color in the format of RGB565 output
LCDC_DMA_MODE_CFG	0x000C	R/W	Configure DMA mode parameters.
LCDC_CLK_DIV	0x0010	R/W	REG and LED clock divider
Interrupt and Status Registers			
LCDC_IRQ_EN	0x0020	R/W	LCDC interrupt enable register
LCDC_IRQ_STATUS	0x0024	R/W	LCDC interrupt status after mask and interrupt clear register

LCDC_IRQ_RAW	0x0028	RO	LCDC raw interrupt register
LCDC_LINE_INT_POS	0x002C	R/W	The position of the line interrupt.
LCDC_CUR_POS_STATUS	0x0030	RO	The position of current display.
LCDC_STATUS	0x0034	RO	Underflow times for debug.
RGB Control Registers			
LCDC_RGB_CFG	0x0040	R/W	RGB I/F signal polarity
LCDC_RGB_VYNC_CFG	0x0044	R/W	VFP/VBP/VSW (Vertical synchronous signal width -1)
LCDC_RGB_HYNC_CFG	0x0048	R/W	HFP/HBP/HSW (Horizontal synchronous signal width -1)
LCDC_RGB_SYNC_STATUS	0x004C	R/W	HSYNC/VSYNC status
MCU Control Registers			
LCDC_MCU_CFG	0x0060	R/W	MCU I/F signal polarity
LCDC_MCU_VSYNC_CFG	0x0064	R/W	MCU VSYNC period under VSYNC mode. MCU VSYNC signal width under VSYNC mode.
LCDC_MCU_TIMING_CFG	0x0068	R/W	The period from CS pulse start to WR pulse start. The hold cycles of RS pulse after first WR pulse released. The period from RS pulse start to WR pulse start.
LCDC_MCU_IO_DATA	0x006C	WO	MCU I/O mode Tx FIFO write MCU I/O mode LCD read
LCDC_MCU_IO_TO_CFG	0x0070	WR	Write clock number configuration for MCU I/O write/read timeout.
LED Control Registers			
LCDC_LED_CFG	0x0080	R/W	Color mode and channels
LCDC_LED_TIMING	0x0084	R/W	LED control timings
LCDC_LED_IDLE	0x0088	R/W	LED line and frame idle timings
Image Control Registers			
LCDC_IMG_BASE_ADDR	0x0090	R/W	Image base address

20.3.1 Global Control Registers

20.3.1.1 LCDC_CTRL

- **Name:** LCDC control register
- **Size:** 32 bits
- **Address offset:** 0x0000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD												LCD_IF_MODE			
												R/W			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD												LCDCINSDIS		LCDCDIS	LCDCEN
												R/W1only		R/W1only	R/W1only

Bit	Name	Access	Reset	Description
31:20	RSVD	N/A	0	Reserved
19:16	LCD_IF_MODE	R/W	0x09	<ul style="list-style-type: none"> ● 0000: 8-bit MCU ● 0001: 16-bit MCU ● 1000: 6-bit RGB parallel interface ● 1001: 16-bit RGB parallel interface ● 1111: LED interface ● Others: Reserved
15:3	RSVD	N/A	0	Reserved
2	LCDCINSDIS	R/W1only	0	Disable LCDC instantly. Reset LCDC internal states and disable LCDC then clear both this bit and LCDCEN. MCU is running CMD (CMDTYPE = 00 or 01) is only terminated with this bit. This bit cannot be set when LCDCDIS is active or at the same time of setting LCDCDIS.

1	LCDCDIS	R/W1only	0	During the period of valid line (VTIMING =valid data), this disable action is performed after the last valid line has transferred. During the other periods, perform the disable action instantly. This disable action resets LCDC internal states and disables LCDC then clear both this bit and LCDCEN. MCU is running CMD (CMDTYPE = 00 or 01) can't be terminated with this bit. This bit cannot be set when LCDCINSDIS active or at the same time of setting LCDCINSDIS.
0	LCDCEN	R/W1only	0	Write 1 to enable LCDC.

20.3.1.2 LCDC_PLANE_SIZE

- **Name:** LCDC plane size register
- **Size:** 32 bits
- **Address offset:** 0x0004
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				IMAGEHEIGHT											
				R/W											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				IMAGEWIDTH											
				R/W											

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	0	Reserved
27:16	IMAGEHEIGHT	R/W	0x140	The height of image (Y-channel based)
15:12	RSVD	N/A	0	Reserved
11:0	IMAGEWIDTH	R/W	0xF0	The width of image (X-channel based) <ul style="list-style-type: none"> ● For LCD, it means pixel number per line. ● For LED, it means LED dot number per line. ● For LED mode, the width must be the multiple of 8.

Note: In MCU and LED I/F mode, the image size must be not less than 32. (IMAGEHEIGHT x IMAGEWIDTH >= 32)

20.3.1.3 LCDC_UNDFLW_CFG

- **Name:** LCDC underflow configuration register
- **Size:** 32 bits
- **Address offset:** 0x0008
- **Read/write access:** read/write

This register defines the underflow color in the format of RGB output. The underflow color is used when DMA FIFO under flow happens. The reset value of 0x00000000 defines a transparent black color.

Note: If a fake pixel value has been send out during underflow, the true pixel data drops. There is an underflow pixel counter to realize this.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD						DMAUNMODE	RSVD								
						R/W									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERROUTDATA															
R/W															

Bit	Name	Access	Reset	Description
31:26	RSVD	N/A	0	Reserved
25	DMAUNMODE	R/W	0	<ul style="list-style-type: none"> ● 0: Output last data

				● 1: Output ERROUTDATA
24:16	RSVD	N/A	0	Reserved
15:0	ERROUTDATA	R/W	0	Output data when DMA FIFO underflow occurred. (directly mapping to output D[15:0]) Note: When DMA under flow happens in LED I/F mode, the least bit of this field is used as the error data.

20.3.1.4 LCDC_DMA_MODE_CFG

- **Name:** LCDC DMA mode configuration register
- **Size:** 32 bits
- **Address offset:** 0x000C
- **Read/write access:** read/write

This register is used to configure DMA mode parameters.

31	30	29	...	10	9	8
DMAINTV						
RO						
7	6	5	4	3	2	1
RSVD			RDOTST		TRIGGER_ONETIME	DMA_TRIGGER_MODE
			R/W		R/W/AC	R/W

Bit	Name	Access	Reset	Description
31:8	DMAINTV	RO	0	The interval cycle count between two DMA requests. (for debug) Unit: bus clock cycle.
7:5	RSVD	N/A	0	Reserved
4:2	RDOTST	R/W	0x7	The value of read outstanding -1, for DMA burst size configuration
1	TRIGGER_ONETIME	R/W/AC	0	Write '1', hardware DMA one frame from DMA buffer based on synchronous signal. After DMA completed, this bit is cleared automatically.
0	DMA_TRIGGER_MODE	R/W	0	DMA Trigger-mode enable. <ul style="list-style-type: none"> ● 0: Auto-mode, open, DMA refreshes automatically based on synchronous signal. ● 1: Trigger-mode open, DMA refreshes frame manually based on TRIGGER_ONETIME bit of this register. Trigger mode is useful for LCD with internal GRAM.

20.3.1.5 LCDC_CLK_DIV

- **Name:** LCDC clock divider register
- **Size:** 32 bits
- **Address offset:** 0x0010
- **Read/write access:** read/write

This register is just for RGB and LED screen.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLKDIV															
R/W															

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	0	Reserved

15:0	CLKDIV	R/W	2	<p>DCLK clock divider depends on this value. The relationship between them is as follows:</p> <ul style="list-style-type: none"> ● 0 : The DCLK clock divider is 2 ● 2 : The DCLK clock divider is 4 ● 4 : The DCLK clock divider is 6 ● 6 : The DCLK clock divider is 8 ● ... <p>The frequency of DCLK is derived from the following equation: $DCLK = SYS_CLK / (CLKDIV + 2)$ where CLKDIV is any even value between 0 and 65534.</p>
------	--------	-----	---	---

20.3.2 Interrupt and Status Registers

20.3.2.1 LCDC_IRQ_EN

- **Name:** LCDC interrupt enable register
- **Size:** 32 bits
- **Address offset:** 0x0020
- **Read/write access:** read/write

31	30	29	...	10	9	8
RSVD						
7	6	5	4	3	2	1
RSVD		FRM_START_INT	IO_TIMEOUT_INTEN	LCD_LIN_INTEN	LCDFRDINTEN	RSVD
		R/W	R/W	R/W	R/W	
						DMAUNINTEN
						R/W

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	0	Reserved
5	FRM_START_INT	R/W	0	<p>DMA Frame start interrupt Enable. This bit can be set and cleared by software.</p> <ul style="list-style-type: none"> ● 0: DMA Frame start interrupt disable ● 1: DMA Frame start interrupt enable
4	IO_TIMEOUT_INTEN	R/W	0	<p>Write or read timeout interrupt Enable in MCU I/F I/O mode. This bit can be set and cleared by software.</p> <ul style="list-style-type: none"> ● 0: MCU I/O mode write/read timeout interrupt disable ● 1: MCU I/O mode write/read timeout interrupt enable
3	LCD_LIN_INTEN	R/W	0	<p>Line Interrupt Enable. This bit can be set and cleared by software.</p> <ul style="list-style-type: none"> ● 0: Line interrupt disable ● 1: Line Interrupt enable
2	LCDFRDINTEN	R/W	0	LCD refresh frame done interrupt enable
1	RSVD	N/A	0	Reserved
0	DMAUNINTEN	R/W	0	DMA FIFO underflow interrupt enable

20.3.2.2 LCDC_IRQ_STATUS

- **Name:** LCDC interrupt status register
- **Size:** 32 bits
- **Address offset:** 0x0024
- **Read/write access:** read/write

LCDC interrupt status after mask and interrupt clear register.

31	30	29	...	10	9	8
RSVD						
7	6	5	4	3	2	1
RSVD		FRM_START_INTS	IO_TIMEOUT_INTS	LCD_LIN_INTS	LCDFRDINTS	RSVD
						DMAUNINTS

		R/W1C	R/W1C	R/W1C	R/W1C		R/W1C
Bit	Name	Access	Reset	Description			
31:6	RSVD	N/A	0	Reserved			
5	FRM_START_INTS	R/W1C	0	DMA frame start interrupt status. Write 1 to clear it. ● 0: No DMA frame start interrupt generated ● 1: A DMA frame start interrupt is generated			
4	IO_TIMEOUT_INTS	R/W1C	0	Write or read timeout interrupt status in MCU I/F I/O mode. Write 1 to clear it. ● 0: No write/read timeout interrupt generated. ● 1: A write/read timeout interrupt is generated.			
3	LCD_LIN_INTS	R/W1C	0	Line Interrupt status. Write 1 to clear it. ● 0: No Line interrupt generated ● 1: A Line interrupt is generated, when a programmed line is reached Note: When this bit is set and DMA fetches the first pixel data of the specified line number from frame buffer, the line interrupt happens. The line number depends on LCDC_LINE_INT_POS register value.			
2	LCDFRDINTS	R/W1C	0	LCD refresh frame done interrupt status. Write 1 to clear it. ● 0: No LCD refresh frame done interrupt generated ● 1: This interrupt generated, when sending LCD frame done. Note: When this bit is set and DMA fetch the last pixel data from frame buffer, the After LCD refresh frame done, interrupt happens.			
1	RSVD	N/A	0	Reserved			
0	DMAUNINTS	R/W1C	0	DMA FIFO underflow interrupt status. Write 1 to clear it. ● 0: No DMA FIFO underflow interrupt generated ● 1: Interrupt generated when DMA FIFO underflow happens			

20.3.2.3 LCDC_IRQ_RAW

- **Name:** LCDC raw interrupt register.
- **Size:** 32 bits
- **Address offset:** 0x0028
- **Read/write access:** read/write

31	30	29	...	10	9	8
RSVD						
7	6	5	4	3	2	1
0	0	0	0	0	0	0
RSVD	FRM_START_INTRS	IO_TIMEOUT_INTRS	LCD_LIN_INTRS	LCDFRDINTRS	RSVD	DMAUNINTRS
	RO	RO	RO	RO		RO

Bit	Name	Access	Reset	Description
31:6	RSVD	N/A	0	Reserved
5	FRM_START_INTRS	RO	0	DMA frame start raw interrupt status. No matter DMA frame start interrupt is enabled or disabled, this bit is set when DMA frame start interrupt happens. Write 1 to the FRM_START_INTRS field in register LCDC_IRQ_STATUS to clear this bit.
4	IO_TIMEOUT_INTRS	RO	0	Write or read timeout interrupt raw interrupt status in MCU I/F I/O mode. No matter the I/O write/read timeout interrupt is enabled or disabled, this bit is set when I/O write/read timeout interrupt happens. Write 1 to the IO_TIMEOUT_INTRS field in register LCDC_IRQ_STATUS to clear this bit.
3	LCD_LIN_INTRS	RO	0	Line Interrupt raw status. No matter the line interrupt is enabled or disabled, this bit is set when line interrupt happens. Write 1 to the LCD_LIN_INTRS field in register LCDC_IRQ_STATUS to clear this bit.
2	LCDFRDINTRS	RO	0	LCD refresh frame done raw interrupt status. No matter the refresh done interrupt is enabled or disabled, this bit is set when refresh done interrupt happens. Write 1 to the LCDFRDINTRS field in register LCDC_IRQ_STATUS to clear this bit.
1	RSVD	N/A	0	Reserved

0	DMAUNINTRS	RO	0	DMA FIFO underflow raw interrupt status. When LCDC DMA FIFO under flow event happens, this bit is set. Write 1 to the DMAUNINTS field in register LCDC_IRQ_STATUS to clear this bit.
---	------------	----	---	--

20.3.2.4 LCDC_LINE_INT_POS

- **Name:** LCDC line interrupt position register
- **Size:** 32 bits
- **Address offset:** 0x002C
- **Read/write access:** read/write

This register defines the position of the line interrupt. The line value to be programmed depends on the timings parameters.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				LINE_INT_POS											
				R/W											

Bit	Name	Access	Reset	Description
31:12	RSVD	N/A	0	Reserved
11:0	LINE_INT_POS	R/W	0	Line Interrupt Position These bits configure the line interrupt position.

20.3.2.5 LCDC_CUR_POS_STATUS

- **Name:** LCDC current position register
- **Size:** 32 bits
- **Address offset:** 0x0030
- **Read/write access:** read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD				CUR_POS_Y											
				RO											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				CUR_POS_X											
				RO											

Bit	Name	Access	Reset	Description
31:28	RSVD	N/A	0	Reserved
27:16	CUR_POS_Y	RO	0	Current Y Position These bits return the current Y position.
15:12	RSVD	N/A	0	Reserved
11:0	CUR_POS_X	RO	0	Current X Position These bits return the current X position.

20.3.2.6 LCDC_STATUS

- **Size:** 32 bits
- **Address offset:** 0x0034
- **Read/write access:** read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAUNINTCNT															

RO

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	0	Reserved
15:0	DMAUNINTCNT	RO	0	DMA FIFO underflow interrupt count

20.3.3 RGB Control Registers

20.3.3.1 LDC_RGB_CFG

- **Name:** LDC RGB configuration register
- **Size:** 32 bits
- **Address offset:** 0x0040
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD					RGB_SYNC_MODE	RGBIFUPDATE	RSVD				DATPL	ENPL	HSPL	VSPL	DCLKPL
					R/W	R/W1S					R/W	R/W	R/W	R/W	R/W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															

Bit	Name	Access	Reset	Description
31:27	RSVD	N/A	0	Reserved
26:25	RGB_SYNC_MODE	R/W	0	<ul style="list-style-type: none"> ● 00: DE mode, frame synchronized with ENABLE signal ● 01: HV mode, frame synchronized with synchronous signal ● Others: Reserved
24	RGBIFUPDATE	R/W1S	0	Force Hardware updates RGB I/F parameters after current LCD refresh frame done. CPU writes 1 to force Hardware updating parameters. After updating, this bit is cleared. When the LDC is running, if the following values related with RGB I/F mode are modified dynamically, only writing 1 to this bit can the newer value be used by hardware after the current frame refresh done. <ul style="list-style-type: none"> ● The CLKDIV field in the LDC_CLK_DIV register ● The VFP, VBP, VSW fields in the LDC_RGB_VSYNC_CFG register ● The HBP, HFP, HSW fields in the LDC_RGB_HSYNC_CFG register
23:21	RSVD	N/A	0	Reserved
20	DATPL	R/W	0	The Data pulse polarity. <ul style="list-style-type: none"> ● 0: Normal ● 1: Inverted
19	ENPL	R/W	1	The ENABLE pulse polarity. <ul style="list-style-type: none"> ● 0: Low level for active data ● 1: High level for active data
18	HSPL	R/W	0	The HSYNC pulse polarity. <ul style="list-style-type: none"> ● 0: Low level synchronous clock ● 1: High level synchronous clock
17	VSPL	R/W	0	The VSYNC pulse polarity. <ul style="list-style-type: none"> ● 0: Low level synchronous clock ● 1: High level synchronous clock
16	DCLKPL	R/W	0	The polarity of the DCLK active edge. <ul style="list-style-type: none"> ● 0: Data fetched at DCLK rising edge ● 1: Data fetched at DCLK falling edge
15:0	RSVD	N/A	0	Reserved

20.3.3.2 LDC_RGB_VSYNC_CFG

- **Name:** LDC RGB vertical synchronization register

- **Size:** 32 bits
- **Address offset:** 0x0044
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD												VFP			
												R/W (SHW)			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD					VBP			RSVD				VSW			
					R/W (SHW)							R/W (SHW)			

Bit	Name	Access	Reset	Description
31:20	RSVD	N/A	0	Reserved
19:16	VFP	R/W (SHW)	3	Front porch line number-1. The number of inactive lines at the end of a frame, before vertical synchronization period.
15:12	RSVD	N/A	0	Reserved
11:8	VBP	R/W (SHW)	3	Back porch line number-1. The number of inactive lines at the start of a frame, after vertical synchronization period.
7:4	RSVD	N/A	0	Reserved
3:0	VSW	R/W (SHW)	1	Vertical synchronization signal width -1. Unit: inactive lines

20.3.3.3 LCDC_RGB_HSYNC_CFG

- **Name:** LCDC RGB horizontal synchronization register
- **Size:** 32 bits
- **Address offset:** 0x0048
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD												HFP			
												R/W (SHW)			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HBP								HSW							
R/W (SHW)								R/W (SHW)							

Bit	Name	Access	Reset	Description
31:24	RSVD	N/A	0	Reserved
23:16	HFP	R/W (SHW)	0x1F	Horizontal front porch -1. The number of DCLK periods between the end of active data and the rising edge of HSYNC. HFP's minimum value is 1
15:8	HBP	R/W (SHW)	0x1F	Horizontal back porch -1. The number of DCLK periods between the falling edge of HSYNC and the start of active data. HBP's minimum value is 1
7:0	HSW	R/W (SHW)	0x17	Horizontal synchronization signal width -1. Unit: DCLK HSW's minimum value is 1

20.3.3.4 LCDC_RGB_SYNC_STATUS

- **Name:** LCDC RGB synchronization status register
- **Size:** 32 bits
- **Address offset:** 0x004C
- **Read/write access:** read-only

This register returns the status of the current display phase which is controlled by the HSYNC, VSYNC, and Horizontal/Vertical DE signals.

Example:

- If the current display phase is the vertical synchronization, the VSSTATUS = 00.
- If the current display phase is the horizontal synchronization, the HSSTATUS = 00.

31	30	29	28	27	26	...	9	8	7	6	5	4	3	2	1	0
RSVD													HSSTATUS		VSSTATUS	
													RO		RO	

Bit	Name	Access	Reset	Description
31:4	RSVD	N/A	0	Reserved
3:2	HSSTATUS	RO	0x10	HSYNC Status. <ul style="list-style-type: none"> 00: HSYNC 01: HFP 10: ACTIVE 11: HBP
1:0	VSSTATUS	RO	0x10	VSYNC Status. <ul style="list-style-type: none"> 00: VSYNC 01: VFP 10: ACTIVE 11: VBP

20.3.4 MCU Control Registers

20.3.4.1 LCDC_MCU_CFG

- **Name:** LCDC MCU configuration register
- **Size:** 32 bits
- **Address offset:** 0x0060
- **Read/write access:** read/write

31	30	29	28	...	19	18	17	16
TEDELAY								
R/W								
15	14	13	12	11	10	9	8	
RSVD				MCU_IO_MODE_RUN	MCU_IO_MODE_EN	MCU_SYNC_MODE		MCUIFUPDATE
				RO	R/W	R/W		R/W1S
7	6	5	4	3	2	1	0	
RSVD		MCUSYPL	TEPL	DATAPL	RDPL	WRPL	RSPL	CSPL
		R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:16	TEDELAY	R/W	0	The delay interval -5. This interval is from detected TE signal to starting frame transfer. Unit: WR pulse width. The maximum value is 65535. The real delay interval is TEDELAY + 4 or TEDELAY + 5.
15:13	RSVD	N/A	0	Reserved
12	MCU_IO_MODE_RUN	RO	0	MCU I/F I/O mode run <ul style="list-style-type: none"> 0: DMA mode run 1: I/O mode run
11	MCU_IO_MODE_EN	R/W	0	MCU I/F I/O mode enable <ul style="list-style-type: none"> 0: Disable I/O mode and open DMA mode after I/O mode FIFO empty. 1: Enable I/O mode after current frame refresh. Poll MCU_IO_MODE_RUN after updating this bit.
10:9	MCU_SYNC_MODE	R/W	1	<ul style="list-style-type: none"> 00: Synchronized with the internal clock 01: Synchronized with VSYNC input 10: Tearing effect line on (Mode1, VSYNC) Others: Reserved
8	MCUIFUPDATE	R/W1S	0	Force Hardware to update MCU I/F Timing shadow register at specific timing. CPU writes 1 to force Hardware updating. After Hardware updating finished, this bit is

				cleared. Software cannot write related shadow registers when MCUIFUPDATE is still active. When the LCDC is running, if the following values related with MCU I/F mode are modified dynamically, only writing 1 to this bit can the newer value be used by hardware after the current frame refresh done. <ul style="list-style-type: none"> ● The TEDELAY field in the LCDC_MCU_CFG register ● The MCUVSW, MCUVSPD fields in the LCDC_MCU_VSYNC_CFG register ● The WRPULW, RDACTW, RDINACTW fields in the LCDC_MCU_TIMING_CFG register
7	RSVD	N/A	0	Reserved
6	MCUSYPL	R/W	0	The MCU VSYNC pulse polarity <ul style="list-style-type: none"> ● 0: Low level for active pulse ● 1: High level for active pulse
5	TEPL	R/W	1	The TE pulse polarity. <ul style="list-style-type: none"> ● 0: Low level for active pulse ● 1: High level for active pulse
4	DATAPL	R/W	0	The Data pulse polarity. <ul style="list-style-type: none"> ● 0: Normal ● 1: Inverted
3	RDPL	R/W	0	The RD pulse polarity. <ul style="list-style-type: none"> ● 0: Data fetched at rising edge ● 1: Data fetched at falling edge
2	WRPL	R/W	0	The WR pulse polarity. <ul style="list-style-type: none"> ● 0: Data fetched at rising edge ● 1: Data fetched at falling edge
1	RSPL	R/W	0	The RS pulse polarity. <ul style="list-style-type: none"> ● 0: Low level for command address ● 1: High level for command parameter
0	CSPL	R/W	0	The CS pulse polarity. <ul style="list-style-type: none"> ● 0: Low level for active pulse ● 1: High level for active pulse

20.3.4.2 LCDC_MCU_VSYNC_CFG

- **Name:** LCDC MCU vertical synchronization register
- **Size:** 32 bits
- **Address offset:** 0x0064
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCUVSPD															
R/W (SHW)															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCUVSPD				RSVD				MCUVSW							
R/W (SHW)								R/W (SHW)							

Bit	Name	Access	Reset	Description
31:12	MCUVSPD	R/W (SHW)	0	<ul style="list-style-type: none"> ● MCUIFMODE = 1: VSYNC idle period. This value equals to VSYNC period – IMAGEWIDTH * IMAGEHEIGHT * WR_pulse_per_pixel-5. Unit: WR pulse width. ● MCUIFMODE = 0: The buffer time between frames. This value equals to the time of a frame's end to the next start -5. Unit: WR pulse width. ● MCUIFMODE = 2: Invalid usage. The maximum value is 1048571.
11:8	RSVD	N/A	0	Reserved
7:0	MCUVSW	R/W (SHW)	0	VSYNC signal width - 1. Unit: WR pulse width. (only for MCU VSYNC mode)

20.3.4.3 LCDC_MCU_TIMING_CFG

- **Name:** LCDC MCU timing register
- **Size:** 32 bits
- **Address offset:** 0x0068
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDINACTW															
R/W (SHW)															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDACTW								WRPULW							
R/W (SHW)								R/W (SHW)							

Bit	Name	Access	Reset	Description
31:21	RDINACTW	R/W (SHW)	0	RD inactive pulse width -1. Unit: system clock
20:10	RDACTW	R/W (SHW)	0	RD active pulse width -1. Unit: system clock
9:0	WRPULW	R/W (SHW)	2	Write clock divider depends on this value. The relationship between them is as follows: <ul style="list-style-type: none"> ● 0: The write clock divider is 2 ● 2: The write clock divider is 4 ● 4: The write clock divider is 6 ● 6: The write clock divider is 8 ● ... The frequency of write clock is derived from the following equation: $\text{Write clock} = \text{SYS_CLK} / (\text{WRPULW} + 2)$ where WRPULW is any even value between 0 and 1022.

20.3.4.4 LCDC_MCU_IO_DATA

- **Name:** LCDC MCU I/O data register
- **Size:** 32 bits
- **Address offset:** 0x006C
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCU_WR_CMD_FLAG		RSVD													
R/W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCU_RW_DATA															
R/W															

Bit	Name	Access	Reset	Description
31	MCU_WR_CMD_FLAG	R/W	0	When writing this register, if this bit is set, the lower 16 bits are command and LCDC sends the lower 16 bits' command to LCM; if this bit is cleared, the lower 16 bits are data and the LCDC sends the lower 16 bits' data to LCM.
30:16	RSVD	N/A	0	Reserved
15:0	MCU_RW_DATA	R/W	0	When writing this register, this lower 16 bits can be command or data and it depends on the bit[31] in this register; when reading this register, this lower 16 bits represent the data read from LCM through MCU I/O mode I/F. Note: Not until the I/O TX FIFO is empty can this register be read. To ensure a correct value is read from this register, the bit IO_TIMEOUT_INTR in LCDC_IRQ_RAW must be being polled until this bit is set (I/O mode TX FIFO empty).

20.3.4.5 LCDC_MCU_IO_TO_CFG

- **Name:** LCDC MCU timeout configuration register

- **Size:** 32 bits
- **Address offset:** 0x0070
- **Read/write access:** read/write

Write/read timeout configuration register in MCU I/F I/O mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IO_TIMEOUT_CLK_NUM															
R/W															

Bit	Name	Access	Reset	Description
31:16	RSVD	N/A	0	Reserved
15:0	IO_TIMEOUT_CLK_NUM	R/W	0x42	The I/O write/read timeout interrupt happens, When writing or reading is timeout for IO_RDY_TO_DOTCLK_NUM system clocks in MCU I/F I/O mode, t. When MCU I/F I/O mode is used, the recommend value is: $16 * WRPULW + RDACTW + RDINACTW$

20.3.5 LED Control Registers

20.3.5.1 LCDC_LED_CFG

- **Name:** LCDC LED configuration register
- **Size:** 32 bits
- **Address offset:** 0x0080
- **Read/write access:** read/write

31	30	29	...		10	9	8
RSVD							
7	6	5	4	3	2	1	0
DAT_PLRTY	OE_PLRTY	LAT_PLRTY	LAT_PLRTY	LEDIFUPDATE	COLORCHAN	COLORNUM	
R/W	R/W	R/W	R/W	R/W1S	R/W	R/W	

Bit	Name	Access	Reset	Description
31:8	RSVD	N/A	0	Reserved
7	DAT_PLRTY	R/W	0	The Data pulse polarity <ul style="list-style-type: none"> ● 0: Normal ● 1: Inverted
6	OE_PLRTY	R/W	0	The OE pulse polarity <ul style="list-style-type: none"> ● 0: Low level for active pulse ● 1: High level for active pulse
5	LAT_PLRTY	R/W	0	The polarity of the LATCH active edge <ul style="list-style-type: none"> ● 0: Data pushed to output buffer at LATCH rising edge ● 1: Data pushed to output buffer at LATCH falling edge
4	LAT_PLRTY	R/W	0	The polarity of the DCLK active edge <ul style="list-style-type: none"> ● 0: Data fetched at DCLK rising edge ● 1: Data fetched at DCLK falling edge
3	LEDIFUPDATE	R/W1S	0	Force Hardware to update LED I/F parameters after current LED refresh frame done. CPU writes 1 to force Hardware updating parameters. After updating, this bit is cleared. When the LCDC is running, if the following values related with LED I/F mode are modified dynamically, only writing 1 to this bit can the newer value be used by hardware after the current frame refresh done. <ul style="list-style-type: none"> ● The LATW, OEACTW bits in the LCDC_LED_TIMING register

				<ul style="list-style-type: none"> The FRMIDLEPD bit in the LCDC_LED_IDLE register
2	COLORCHAN	R/W	0	<ul style="list-style-type: none"> 0: One channel 1: Two channels
1:0	COLORNUM	R/W	0x0	Color number in frame buffer (SRAM/PSRAM) <ul style="list-style-type: none"> 0: One color 1: Two colors 2: Three colors 3: Reserved

20.3.5.2 LCDC_LED_TIMING

- Name:** LCDC LED timing register
- Size:** 32 bits
- Address offset:** 0x0084
- Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD								OEACTW							
								R/W							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEACTW								LATW							
R/W								R/W							

Bit	Name	Access	Reset	Description
31:24	RSVD	N/A	0	Reserved
23:8	OEACTW	R/W	0	OE Active Width Time – 1 (unit: dotclock).
7:0	LATW	R/W	0	LAT Width Time - 1 (unit: dotclock).

20.3.5.3 LCDC_LED_IDLE

- Name:** LCDC LED idle register
- Size:** 32 bits
- Address offset:** 0x0088
- Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FRMIDLEPD												RSVD			
R/W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								LINEIDLEPD							
								R/W							

Bit	Name	Access	Reset	Description
31:20	FRMIDLEPD	R/W	0	Frame idle period interval – 1 (unit: dotclock).
19:8	RSVD	N/A	0	Reserved
7:0	LINEIDLEPD	R/W	0	Line idle period time – 1 (unit: dotclock).

20.3.6 Image Control Registers

20.3.6.1 LCDC_IMG_BASE_ADDR

- Name:** LCDC image base address register
- Size:** 32 bits
- Address offset:** 0x0090
- Read/write access:** read/write

31	30	29	28	27	26	25	...	6	5	4	3	2	1	0
IMG_BASE_ADDR														
R/W														

Bit	Name	Access	Reset	Description
31:0	IMG_BASE_ADDR	R/W	0	Image DMA source address. After a frame refresh done, hardware loads the newer base address from this register automatically.

20.4 Programming the LCDC

Table 20-4 lists the typical application scenarios of LCDC.

Table 20-4 Typical application scenario of LCDC

I/F	Data Mode	LCD GRAM	Ameba-D Frame Buffer
RGB	DMA Auto-mode	No	Yes
MCU	DMA Trigger-mode	Yes	Yes
	I/O mode	Yes	No

20.4.1 RGB DMA Auto-mode

- Disable LCDC by writing 0 to the LCDCEN bit of LCDC_CTRL register.
- Configure LCD via SPI if necessary. (Synchronous related parameters: HBP/HFP/HSW/VBP/VFP/VSW)
- Set LCDC_PLANE_SIZE register.
- Set LCDC_CTRL register as RGB interface.
- Set LCDC_DMA_MODE_CFG register as DMA Auto-mode.
- Set LCDC_IMG_BASE_ADDR register for DMA buffer addresses.
- Configure RGB I/F registers for this transfer.
 - Set LCDC_RGB_CFG register as preferred RGB interface profile.
 - Set LCDC_RGB_VSYNC_CFG & LCDC_RGB_HSYNC_CFG register for HBP/HFP/HSW/VBP/VFP/VSW.
- Enable necessary interrupts.
- Start transfer by writing 1 to the LCDCEN bit of LCDC_CTRL register.
- Copy image to DMA buffer, you can get fresh status if needed.
- The transfer is stopped when disabled LCDC.

20.4.2 MCU DMA Trigger-mode

- Disable LCDC by writing 0 to the LCDCEN bit of LCDC_CTRL register.
- Set LCDC_PLANE_SIZE register.
- Set LCDC_CTRL register as MCU interface.
- Set LCDC_DMA_MODE_CFG register as DMA Trigger-mode.
- Configure MCU I/F registers for this transfer.
 - Set LCDC_MCU_CFG register as preferred MCU profile.
 - Set LCDC_MCU_TIMING_CFG register for MCU timings.
 - Set LCDC_MCU_CFG register to enable MCU I/O mode.
- Start transfer by writing 1 to the LCDCEN bit of LCDC_CTRL register.
- Send CMD & CMD parameters to LCD through LCDC_MCU_IO_DATA register.
- Set LCDC_IMG_BASE_ADDR register for DMA buffer addresses.
- Set LCDC_MCU_CFG registers to enable MCU DMA mode (ex. LCD initialization and so on).
- Update image to DMA buffer, you can get fresh status if needed.
- Trigger one time DMA through LCDC_DMA_MODE_CFG registers.

20.4.3 MCU I/O Mode

- (1) Disable LCDC by writing 0 to the LCDCEN bit of LCDC_CTRL register.
- (2) Set LCDC_PLANE_SIZE register.
- (3) Set LCDC_CTRL register as MCU interface.
- (4) Configure MCU I/F registers for this transfer.
 - a) Set LCDC_MCU_CFG register as preferred MCU profile.
 - b) Set LCDC_MCU_TIMING_CFG register for MCU timings.
 - c) Set LCDC_MCU_CFG register to enable MCU I/O mode.
- (5) Start transfer by writing 1 to the LCDCEN bit of LCDC_CTRL register.
- (6) Send CMD & CMD parameters to LCD through LCDC_MCU_IO_DATA register.
- (7) Write/read point to/from GRAM through I/O mode.

21 Quadrature Decoder (Q-Decoder)

21.1 Overall Description

21.1.1 Introduction

Quadrature encoders are used to determine the position and speed of a rotary device, such as servo-motors, volume control wheels, PC mice. The decoded quadrature signals are used as a sensory input to a system to determine the absolute or relative position of the rotary device, which can be used in a control loop, for example, the servo-motor.

21.1.2 Features

- 16-bit position counter
- Index signal of rotation
- Position compares interrupt
- Rotation compares interrupt
- Automatic initialization
- Input debounce
- Auto-index mechanism
- Rotation velocity measurement
- Velocity compare function with “less or more than” interrupt
- The maximum input signal frequency is 200kHz, or 100kHz when the index signal related functions are used.

21.1.3 Application Scenario

Fig 21-1 shows typical quadrature signals from a rotary encoder. The signals named PHA and PHB are two quadrature signals. The figure shows how the phase relationship determines the direction of rotation. When PHA leads PHB, the rotation is defined as forward (+). When PHB leads PHA, the rotation is defined as reverse (-). The concatenation of the two phase signals is called the quadrature state or the phase state.

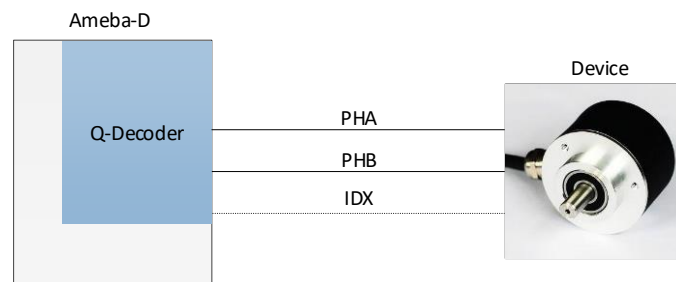


Fig 21-1 Q-Decoder application scenario

The signal shown in Fig 21-1 as IDX (index) is used for absolute position. It is usually tagged as “0” of absolute position. The width of an IDX pulse must be smaller than a 4-state cycle. If the width of the IDX pulse is bigger than one state, the software can use the register setting to locate the IDX pulse to a specified state. Therefore, the Q-Decoder hardware can process the IDX pulse signal only located in the specified state. The IDX pulse signal is located at state “00” in the example shown in Fig 21-2.

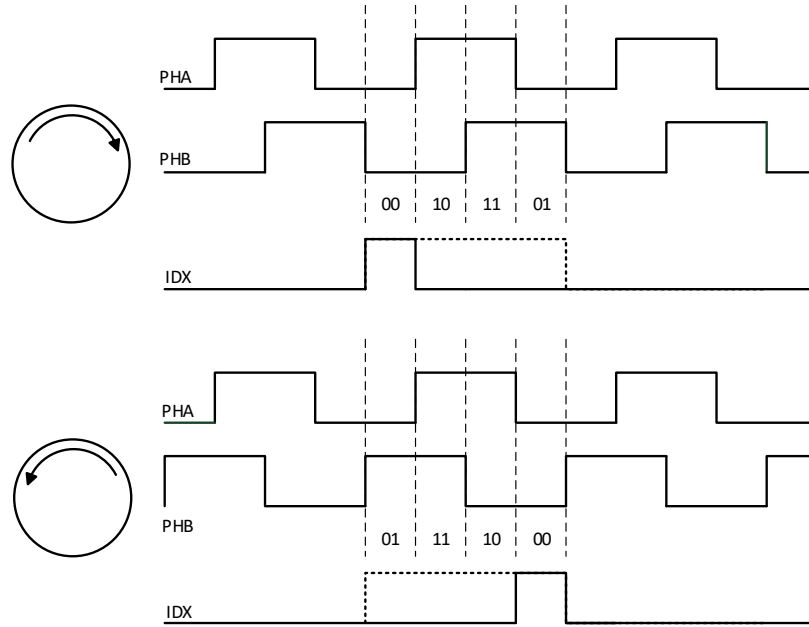


Fig 21-2 Quadrature signal

21.2 Architecture

21.2.1 Q-Decoder Block Diagram

The block diagram of Q-Decoder is show in Fig 21-3.

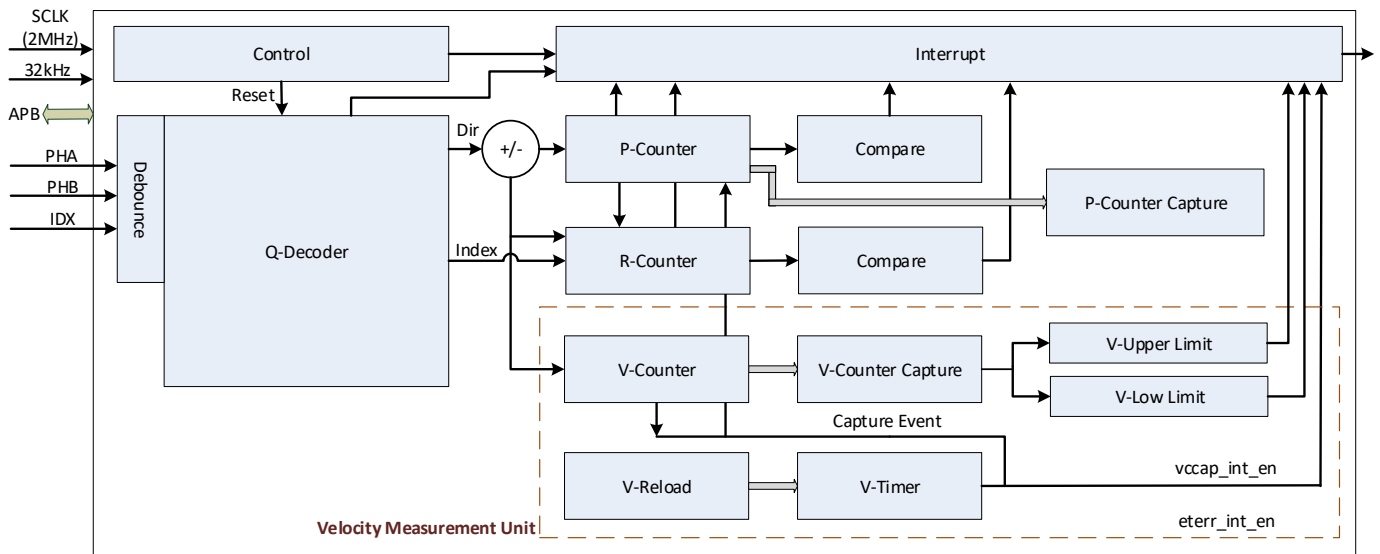


Fig 21-3 Q-Decoder block diagram

21.2.2 Position Measurement

21.2.2.1 Position Counter Measurement

The Q-Decoder is designed to decode the quadrature signals from I/O pin. It can generate the position counter (PC) and the direction by the state changes of the signal PHA and PHB. Therefore, the software can get the position by reading the related registers. Fig 21-4 shows how a Q-Decoder counts the position and detects the direction by PHA and PHB.

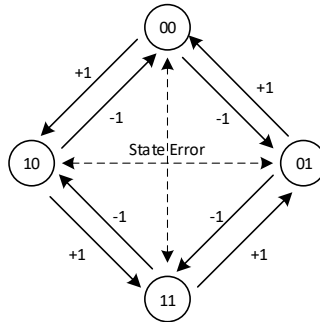


Fig 21-4 Q-Decoder phase state

In the example of Fig 21-5 and Fig 21-6, the maximum position counter (MPC) is set to 199.

- If the movement direction is forward and the position counter is equal to MPC, the PC is reset to 0 on the next state.
- If the movement direction is reverse and the position counter is 0, the PC is reset to MPC on the next state.

The difference between these two examples is CNT_SC. CNT_SC bit is used to configure the number of the phase state changed for the position accumulation counter to be increased/decreased by 1. If CNT_SC = 0, position counter increases/decreases on every phase state change. If CNT_SC = 1, only PHA changes can trigger position counter change.

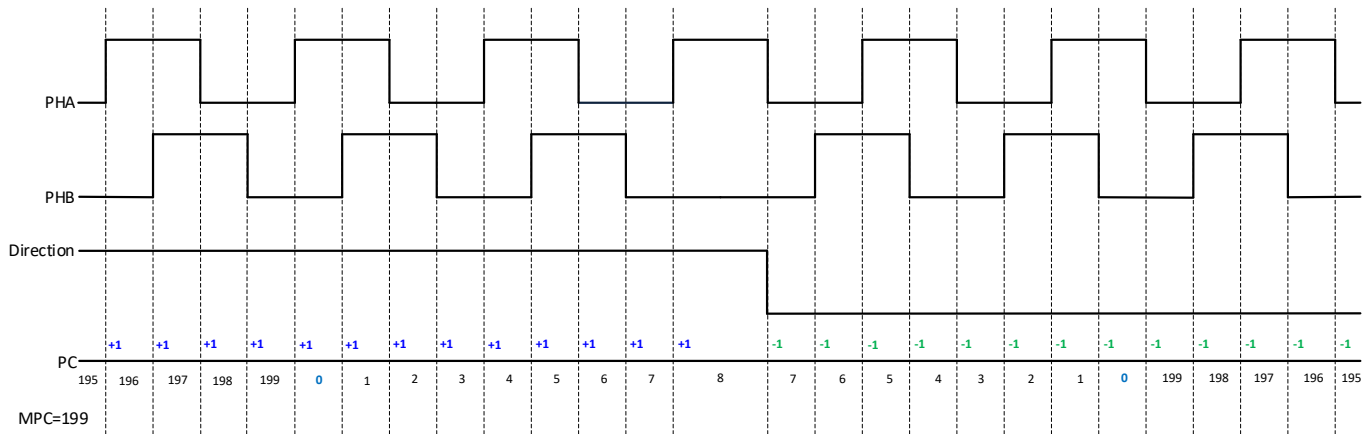


Fig 21-5 Position count state when CNT_SC = 0

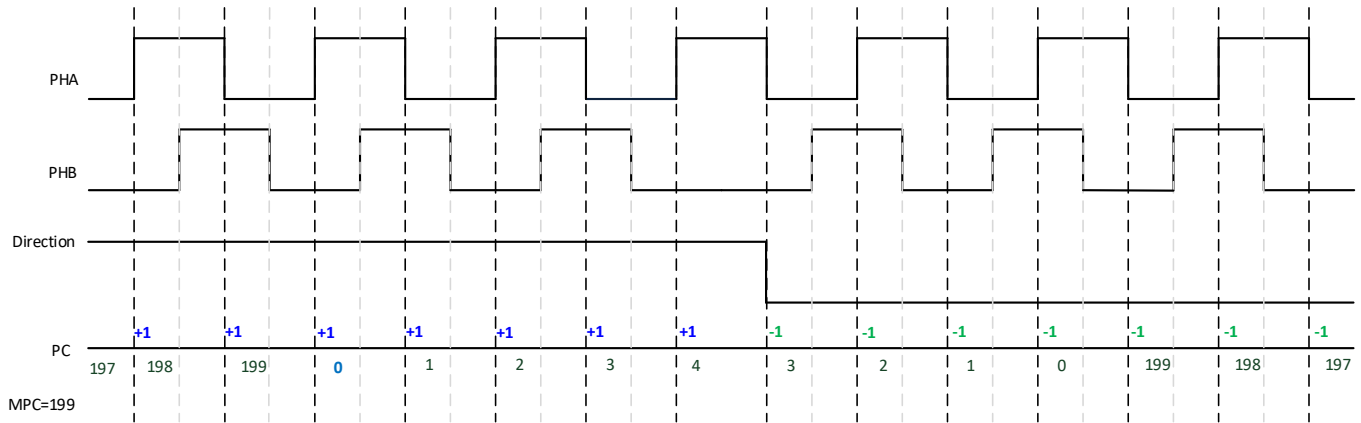


Fig 21-6 Position count state when CNT_SC = 1

21.2.2.2 Position Counter Reset Mechanism

For applications that needs the absolute position, the position counter needs to be reset on the index pulse. Fig 21-7 and Fig 21-8 shows the position counter reset mechanism of the Q-Decoder. In the example of Fig 21-7, the movement direction is forward and the position counter is reset on the index pulse signal with the (PHA, PHB) state is (0, 0). In the example of Fig 21-8, the movement direction is reverse and the position counter is reset on the index pulse signal with the (PHA, PHB) state is (0, 0). Besides, the maximum position counter is set to 199.

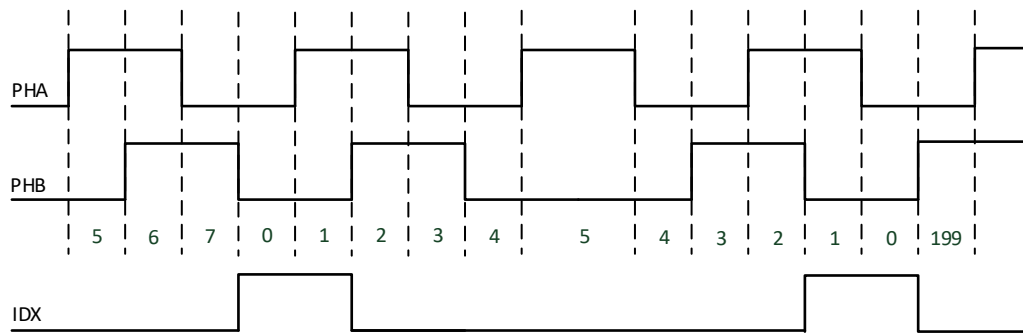


Fig 21-7 Position counter reset on index pulse (forward direction)

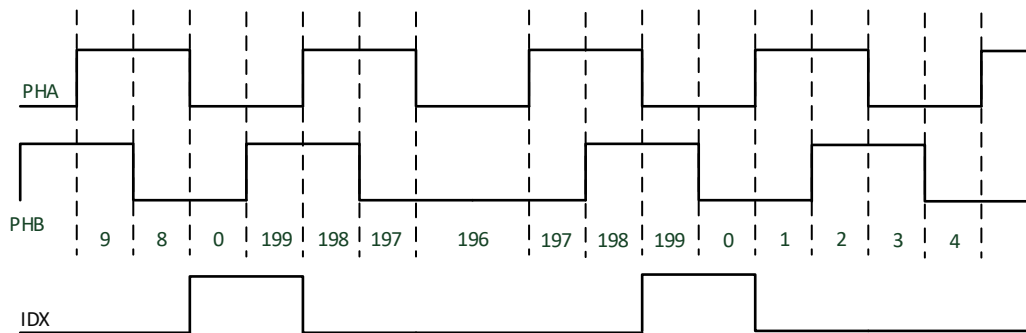


Fig 21-8 Position counter reset on index pulse (reverse direction)

Because of processing technology, for some encoder, the index pulse can not be fully aligned with the edge of state phase changing. If the width of the index pulse is bigger than 1 state, the software can use the register setting to locate the index pulse to a specified state and reset the position counter.

In the example of Fig 21-9, position counter is reset on state (1, 0). In this case, the position counter can also configured to be reset on state (1, 1), (0, 1), but state (0, 0) fails. For some encoder, index outputs high in idle states. In this case, software can configure `IDX_INV` to suite different encoder.

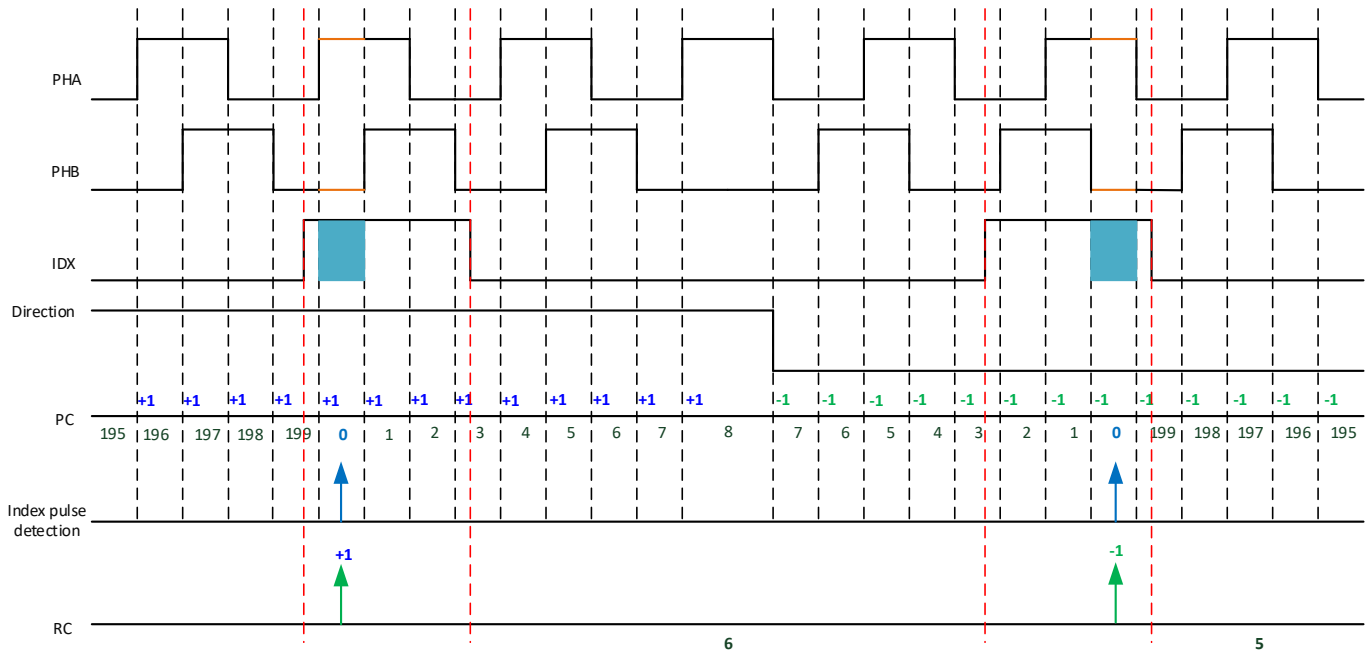


Fig 21-9 `IDX_INV = 0`, position counter is reset on (PHA, PHB) = (1, 0)

Fig 21-10 illustrates the case of inverse index pulse.

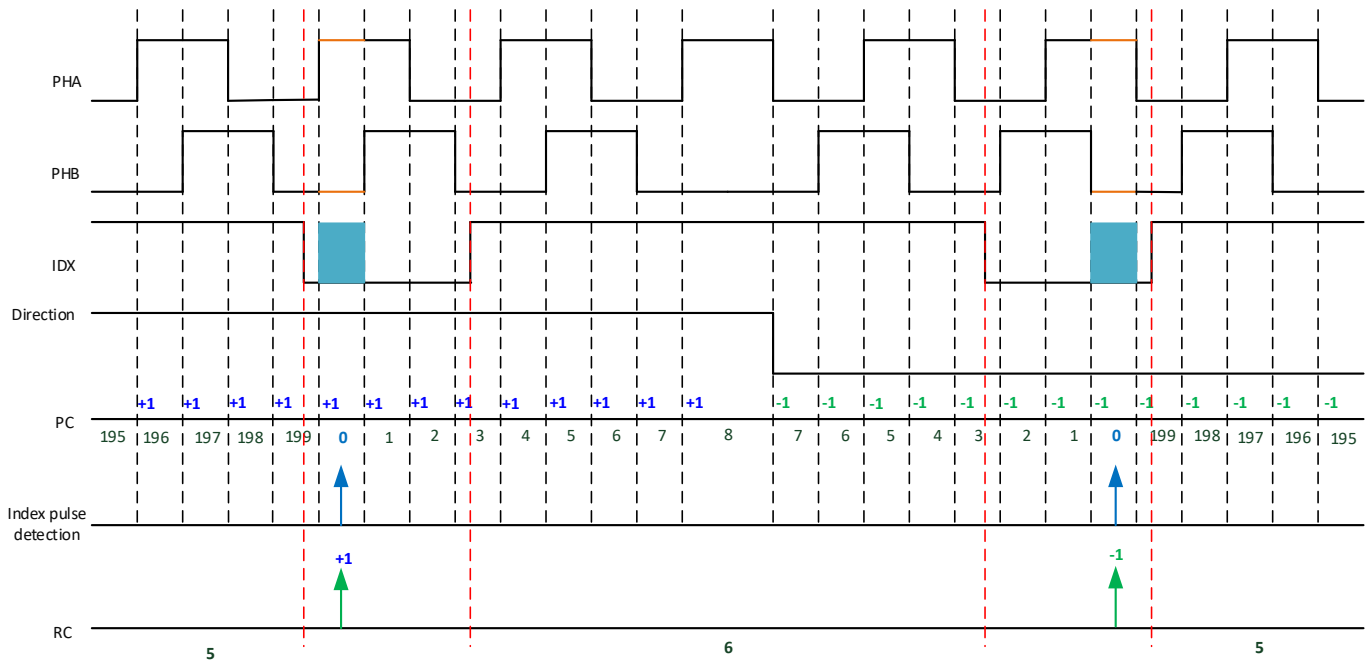


Fig 21-10 `IDX_INV = 1`, position counter is reset on (PHA, PHB) = (1, 0)

Ameba-D also supports auto-index mechanism, which means it can directly execute the index functions without any index setting. This auto-index mechanism is related with index phase.

- When the default phase is low, the index pulse detection is the rising edge in the positive direction, and the falling edge in the negative direction.
- When the default phase is high, the index pulse detection is the falling edge in the positive direction, and the rising edge in the negative direction.

Getting the pulse detection can check the position value, reset the position value, and count the rotation in the example of Fig 21-11 and Fig 21-12.

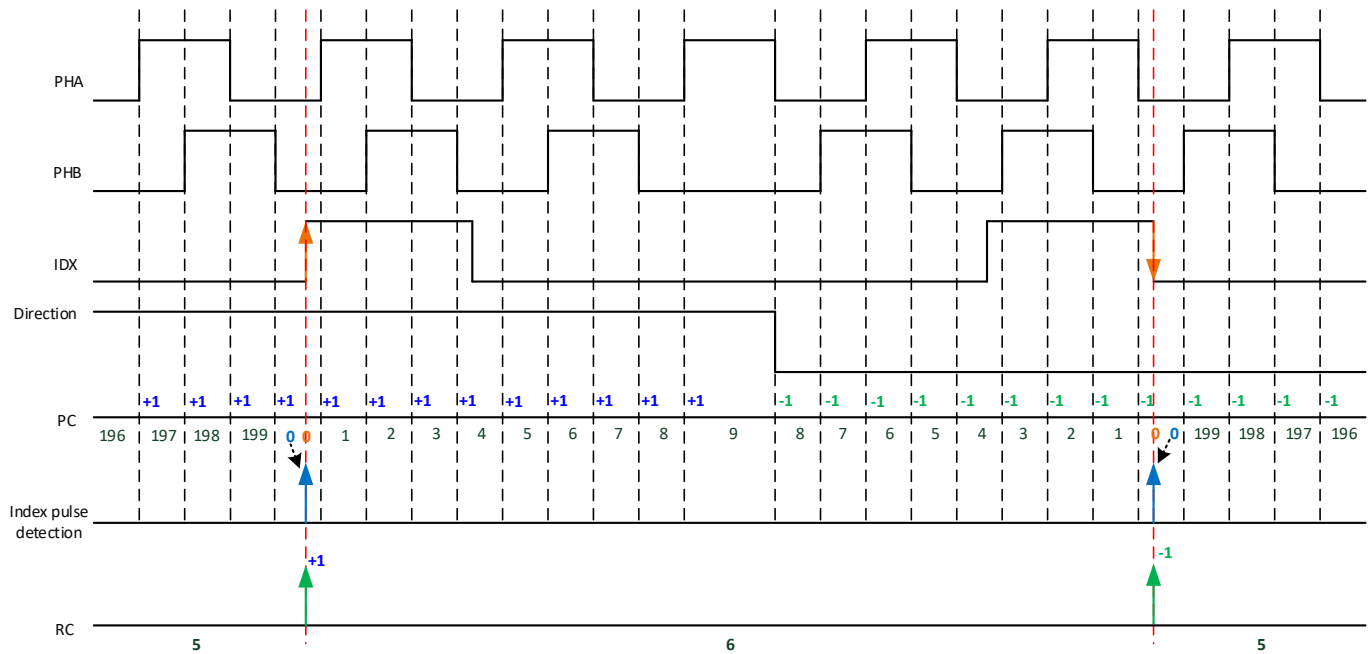


Fig 21-11 Auto-index mechanism when $IDX_INV = 0$

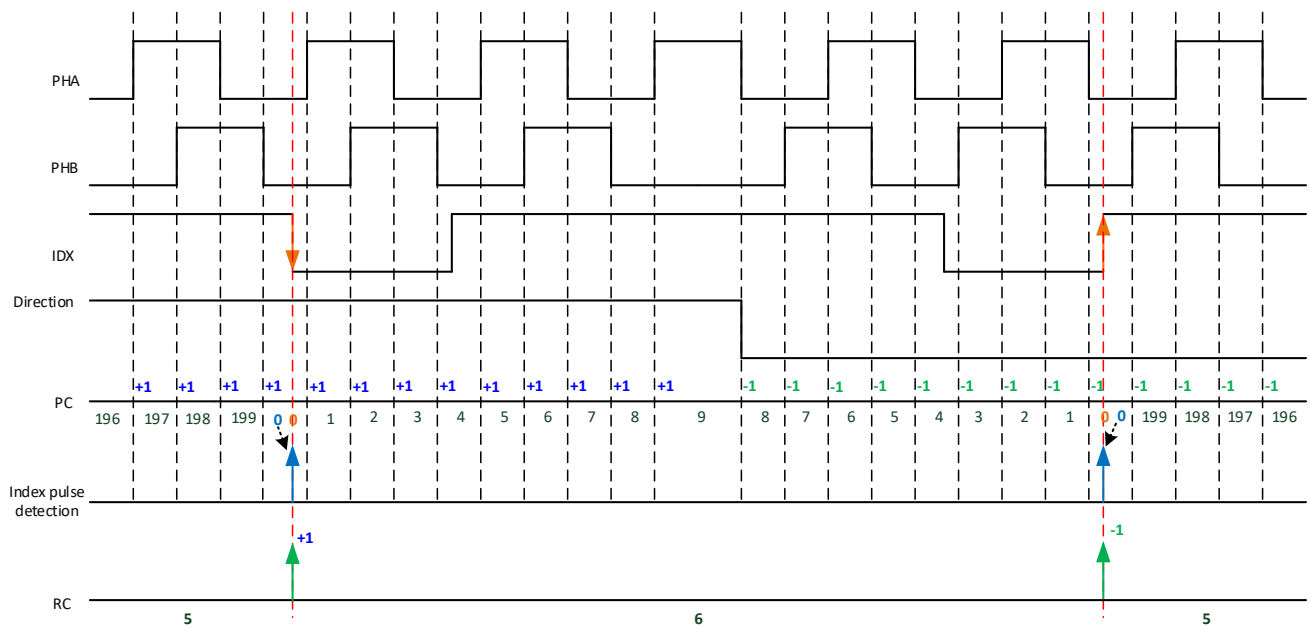


Fig 21-12 Auto-index mechanism when $IDX_INV = 1$

To avoid unpredictable PC value, the time difference between the related edge (can be different due to the direction and default phase mentioned above) of index signal and the edge of neighboring PHA or PHB signal must be larger than 1 sampling clock time.

Note: You should insert a time delay operation between a subsequent reset operation and a disabled Q-Decoder operation. The delay time period must be larger than 2 * divided sampling clock time.

21.2.2.3 Rotation Counter Measure

For rotation device that needs absolute position, it is not intuitive enough to indicate the position using a position counter. A rotation counter is designed to solve this problem.

- If the index pulse signal is enabled (depending on IDX_EN) and the rotation counter mode is 0 (depending on RC_MOD), this rotation counter increases or decreases (depending on the direction) by 1 on every index pulse. Fig 21-9 to Fig 21-12 give some cases.
- If the index pulse signal is disabled (depending on IDX_EN) or the rotation counter mode is 1 (depending on RC_MOD), this rotation counter increases or decreases by 1 when every position counter overflow (+1) or underflow (-1) occurs. Fig 21-13 illustrates this case.

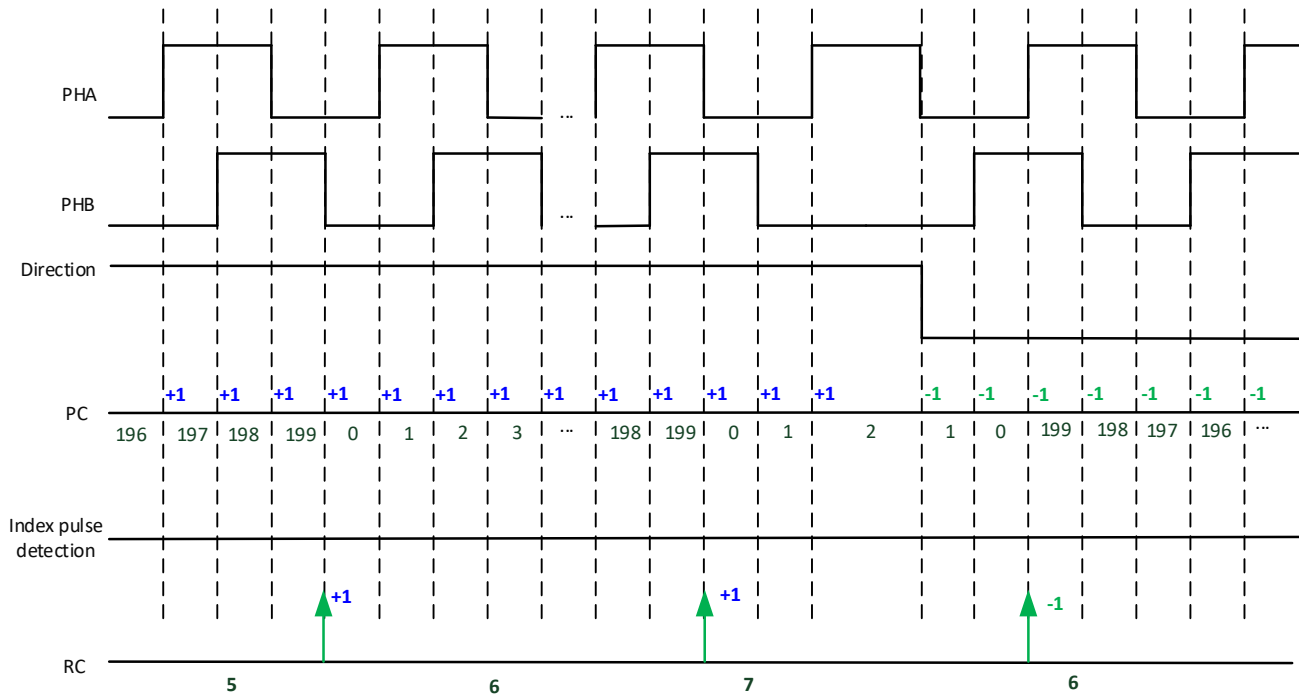


Fig 21-13 Rotation count when RC_MOD = 1

21.2.3 Velocity Measurement

For some applications, for example, a flow meter, the software needs to use the position information with a time period to calculate the device rotation speed.

The speed can be calculated with the following equation.

$$V \approx \Delta X / T \approx V\text{-counter} / V\text{-Reload}$$

$$\approx [P\text{-counter capture}(t) - P\text{-counter capture}(t-1)] / V\text{-Reload}$$

This equation is to measure the position difference between the zero time events of the velocity timer. The position difference can be obtained from the velocity counter or subtracts two position counter capture values.

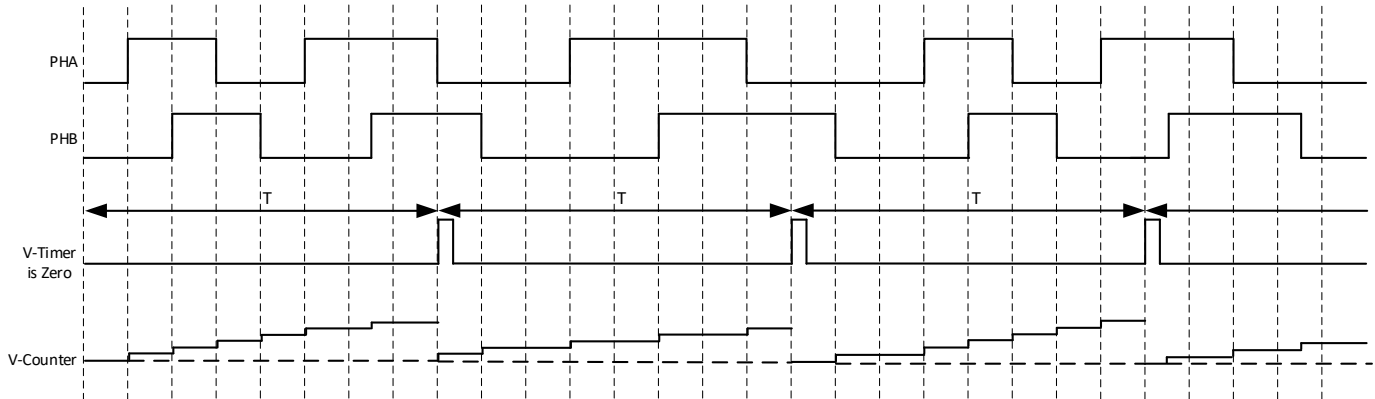


Fig 21-14 Velocity measurement unit timing flow

21.3 Registers

Table 21-1 lists the details of Q-Decoder registers.

Table 21-1 Q-Decoder registers

Name	Address Offset	Access	Description
Global Control Registers			
REG_CLK_SEL	0x0000	R/W	Clock Configuration Register
REG_CTRL	0x0004	R/W	Q-Decoder Control Register
Position Measurement Registers			
REG_MPC	0x0008	R/W	Q-Decoder Max Position Counter Register
REG_RC	0x000C	R/W	Q-Decoder Rotation Compare Register
REG_PC	0x0010	RO	Q-Decoder Position Counter Register
REG_ISC	0x0014	R/W	Q-Decoder Index Signal Configuration Register
Velocity Measurement Registers			
REG_VCTRL	0x0018	R/W	Q-Decoder Velocity Control Register
REG_VC	0x001c	RO	Q-Decoder Velocity Counter Register
REG_VCCAP	0x0020	RO	Q-Decoder Velocity Counter Capture Register
REG_PCCAP	0x0024	RO	Q-Decoder Position Counter Capture Register
REG_VTRLD	0x0028	R/W	Q-Decoder Velocity Time Reload Register
REG_VT	0x002c	R/W	Q-Decoder Velocity Timer Register
REG_VCOMP	0x0030	R/W	Q-Decoder Velocity Compare Register
Interrupt Registers			
REG_IMR	0x003C	R/W	Q-Decoder Interrupt Mask Register
REG_ISR	0x0040	R/W	Q-Decoder Interrupt Status Register

21.3.1 Global Control Registers

21.3.1.1 REG_CLK_SEL

- **Name:** Clock Configuration Register
- **Size:** 32 bits
- **Address offset:** 0x0000
- **Read/write access:** read/write

31	30	29	...	19	18	17	16	15	14	13	12	11	10	9	..	1	0
RSVD							SMP_DIV					RSVD	DBN_TM				
							R/W						R/W				

Bit	Name	Access	Default	Description
31:17	RSVD	N/A	-	Reserved
16:12	SMP_DIV	R/W	0x0	Divider for input signal sampling clock Sampling Clock = source-clock / (SMP_DIV + 1)
11	RSVD	N/A	-	Reserved
10:0	DBN_TM	R/W	0	Debounce timer configuration The debounce time period is DBN_TM * one sampling clock time, sampling clock can be 32.768kHz or 2MHz.

21.3.1.2 REG_CTRL

- **Name:** Q-Decoder Control Register
- **Size:** 32 bits
- **Address offset:** 0x0004
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
AXIS_EN	PC_RST	RC_RST	RC_MOD	QALL_RST	RSVD		RC_INT_EN
R/W	R/W	R/W	R/W	R/W			R/W
23	22	21	20	19	18	17	16
PCE_INT_EN	IDX_INT_EN	RUF_INT_EN	ROF_INT_EN	PC_INT_EN	DR_INT_EN	CT_INT_EN	OF_INT_EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
15	14	13	12	11	10	9	8
UF_INT_EN	IL_INT_EN	CNT_SC	DBN_EN	RSVD			
R/W	R/W	R/W	R/W				
7	6	5	4	3	2	1	0
RSVD			PCHG_LV		MNU_INI	INI_PHASE	
			R/W		R/W	R/W	

Bit	Name	Access	Default	Description
31	AXIS_EN	R/W	0	Q-Decoder enable control <ul style="list-style-type: none"> ● 1: Enable ● 0: Disable When the CPU writes 0 to this bit, hardware works as follows: <ul style="list-style-type: none"> ● Source clock is gated. ● Clear all interrupt status.
30	PC_RST	R/W	0	Position counter reset <ul style="list-style-type: none"> ● 0: Position counter is not in reset state (normal active). ● 1: Writing 1 to this bit resets the position counter to 0. The position counter reset can only be performed when the Q-Decoder is disabled (AXIS_EN = 0). Writing 1 to this bit while the Q-Decoder is enabled (AXIS_EN = 1) does not reset the position counter.
29	RC_RST	R/W	0	Rotation counter reset <ul style="list-style-type: none"> ● 0: Rotation counter is not in reset state. ● 1: Writing 1 to this bit resets the rotation counter to 0. The rotation counter reset can only be performed when Q-Decoder is disabled (AXIS_EN = 0). Writing 1 to this bit while Q-Decoder is enabled does not reset the rotation counter.
28	RC_MOD	R/W	0	Rotation counter mode <ul style="list-style-type: none"> ● 0: The rotation counter is used to accumulate the number of index event occurred with direction (+/-). The criterion of determining an index event occurrence is the same as the index reset according to auto reset or index reset setting. ● 1: The rotation counter is used to accumulate the number of the position counter overflow (+)/underflow (-).
27	QALL_RST	R/W	0	Quadrature decoder all reset <ul style="list-style-type: none"> ● 0: Quadrature decoder is not in reset state. ● 1: Writing 1 to this bit resets the state machine and resets all functions. The Q-Decoder reset can only be performed when the Q-Decoder is disabled.

26:25	RSVD	N/A	-	Reserved
24	RC_INT_EN	R/W	0	<p>Rotation counter comparing interrupt enable control</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt <p>This interrupt is asserted when the rotation counter is equal to the value of 'RCC'.</p>
23	PCE_INT_EN	R/W	0	<p>Position counter error interrupt enable control</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt <p>This interrupt is asserted when the index pulse signal is detected but the position counter is not equal to 0.</p> <p>This bit is valid only when the index pulse detection is enabled (depending on IDX_EN).</p>
22	IDX_INT_EN	R/W	0	<p>Index pulse signal interrupt enable control</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt <p>This interrupt is asserted when the index pulse signal presents.</p> <p>This bit is valid only when the index pulse detection is enabled (depending on IDX_EN).</p>
21	RUF_INT_EN	R/W	0	<p>Rotation counter underflow interrupt enable control</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt <p>This interrupt is asserted when the rotation counter underflow occurs (0 → 0xFFF).</p>
20	ROF_INT_EN	R/W	0	<p>Rotation counter overflow interrupt enable control</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt <p>This interrupt is asserted when the rotation counter overflow occurs (0xFFF → 0).</p>
19	PC_INT_EN	R/W	0	<p>Position counter comparing interrupt enable control</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt <p>This interrupt is asserted when the position counter is equal to the value of 'PCC'.</p>
18	DR_INT_EN	R/W	0	<p>Movement direction changed interrupt enable control:</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt <p>When the movement direction changes, this interrupt is asserted. Keep the direction at the next state change if the Q-Decoder is reset (perform QALL_RST or power on reset).</p>
17	CT_INT_EN	R/W	0	<p>Position counter value changed interrupt enable control</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt <p>When the position counter value changes and the difference (base on initial value or previous interrupt triggered value) is over the level (depending on PCHG_LV), the position changed interrupt is asserted.</p>
16	OF_INT_EN	R/W	0	<p>Position counter value overflow (max position counter → 0x0000) interrupt enable control</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt
15	UF_INT_EN	R/W	0	<p>Position counter value underflow (0x0000 → max position counter) interrupt enable control</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt
14	IL_INT_EN	R/W	0	<p>Illegal state (phase A and phase B toggle simultaneously) detected interrupt enable control</p> <ul style="list-style-type: none"> ● 1: Enable interrupt ● 0: Disable interrupt
13	CNT_SC	R/W	0	<p>This bit is used to configure the number of the phase state changed for the position accumulation counter to be increased/decreased by 1.</p> <ul style="list-style-type: none"> ● 0: 1 phase. ● 1: 2 phases. Only phase A edges are counted.
12	DBN_EN	R/W	0	<p>Input signal (PHA, PHB & IDX) debouncing enable control</p> <ul style="list-style-type: none"> ● 1: Enable debounce ● 0: Disable debounce
11:5	RSVD	N/A	-	Reserved

4:3	PCHG_LV	R/W	0	Position changed interrupt trigger level This field sets a value changed on the position counter, which triggers the position changed interrupt. <ul style="list-style-type: none"> 00: +/- 1 01: +/- 2 10: +/- 4 11: Reserved (never set 0x11 to these bits)
2	MNU_INI	R/W	0	This bit is used to enable the function of initializing the phase state of the Q-Decoder manually. <ul style="list-style-type: none"> 1: Manual set (software may read the state via GPIO and then write it to 'INI_PHASE') 0: Auto load when the Q-Decoder is enabled; only trigger the auto load after rising edge of 'AXIS_EN' = 1.
1:0	INI_PHASE	R/W	0	The manual initial (A, B) phase state of the quadrature decoder <ul style="list-style-type: none"> 00: (0, 0) 01: (0, 1) 10: (1, 0) 11: (1, 1) This field is valid only when the manually initial phase state function is enabled (MNU_INI = 1).

21.3.2 Position Measurement Registers

21.3.2.1 REG_MPC

- Name:** Q-Decoder Max Position Counter Register
- Size:** 32 bits
- Address offset:** 0x0008
- Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCC															
R/W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPC															
R/W															

Bit	Name	Access	Default	Description
31:16	PCC	R/W	0	The value to be compared with the position counter. If the position counter is equal to this value, the corresponding interrupt (pc_cmp) is asserted.
15:0	MPC	R/W	0xFFFF	The maximum value of the position counter. <ul style="list-style-type: none"> If the position counter is equal to this value and the movement direction is forward, the position counter is reset to 0 on the next counter increment, and the position counter overflow interrupt also occurs. If the position counter is equal to 0 and the movement direction is backward, the position counter is reset to this value on the next counter decrement, and the position counter underflow interrupt also occurs.

21.3.2.2 REG_RC

- Name:** Q-Decoder Max Rotation Compare Register
- Size:** 32 bits
- Address offset:** 0x000C
- Read/write access:** read/write

31	30	29	28	...	15	14	13	12	11	10	9	...	2	1	0
RSVD									RCC						
									R/W						

Bit	Name	Access	Default	Description
31:12	RSVD	N/A	-	Reserved
11:0	RCC	R/W	0	The value to be compared with the rotation counter. If the rotation counter is equal to this value, the corresponding interrupt is asserted.

21.3.2.3 REG_PC

- **Name:** Q-Decoder Position Counter Register
- **Size:** 32 bits
- **Address offset:** 0x0010
- **Read/write access:** read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RC												STA		ALS	DIR
R												R		R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC															
R															

Bit	Name	Access	Default	Description
31:20	RC	R	0	<p>The rotation counter</p> <ul style="list-style-type: none"> ● If the index pulse signal is enabled (depending on IDX_EN) and the rotation counter mode is 0 (depending on RC_MOD), this rotation counter increases or decreases (depending on the direction, DIR) by 1 on every index pulse. ● If the index pulse signal is disabled (depending on IDX_EN) or the rotation counter mode is 1 (depending on RC_MOD), this rotation counter increases or decreases by 1 for every position counter overflow (+1) or underflow (-1) occurs. <p>When the rotation counter overflow or underflow occurs, a corresponding interrupt is asserted.</p>
19:18	STA	R	0	The Q-Decoder phase state, current state of (A, B) phase.
17	ALS	R	0	<p>The status of auto load to set initial phase</p> <ul style="list-style-type: none"> ● 1: Auto-load is done ● 0: Auto-load is ongoing <p>If the initial phase auto-load is disabled, the value of this bit should be ignored.</p>
16	DIR	R	0	<p>The movement direction</p> <ul style="list-style-type: none"> ● 0: Decrease ● 1: Increase
15:0	PC	R	0	<p>The position accumulation counter</p> <p>Step is +1, -1 or reset to 0.</p> <p>This position counter increases or decreases (depending on the direction) by 1 for every 1 or 2 (depending on CNT_SC) phase states change.</p> <ul style="list-style-type: none"> ● If this position counter is equal to maximum position counter (MPC) and the movement direction is forward, it is reset to 0 on the next counter increment. ● If this position counter is equal to 0 and the movement direction is backward, it is reset to the maximum position counter (MPC) on the next counter decrement. <p>If the position counter reset on index pulse is enabled (depending on POS_RST_EN), this position counter is reset to 0 on the index pulse signal.</p>

21.3.2.4 REG_ISC

- **Name:** Q-Decoder Index Signal Configuration Register
- **Size:** 32 bits
- **Address offset:** 0x0014
- **Read/write access:** read/write

31	30	29	28	...	9	8	7	6	5	4	3	2	1	0
IDX_EN	RSVD							IDX_AUTO_EN	IDX_INV	POS_RST_EN	RSVD		POS_RST_PHA	POS_RST_PHB
R/W								R/W	R/W	R/W			R/W	R/W

Bit	Name	Access	Default	Description
31	IDX_EN	R/W	0	Enable the index pulse detection <ul style="list-style-type: none"> 0: Disable. The detection of index pulse signal is disabled, all index pulse signal related configurations should be ignored. 1: Enable.
30:7	RSVD	N/A	-	Reserved
6	IDX_AUTO_EN	R/W	0	Auto-index mechanism <ul style="list-style-type: none"> 0: Disable. 1: Enable. When enable this function, ignore the setting pos_rst_mode, pos_rst_pha, and POS_RST_PHB.
5	IDX_INV	R/W	0	Inverse the index pulse input signal <ul style="list-style-type: none"> 0: No inverse 1: Inverse the index pulse signal input
4:3	POS_RST_EN	R/W	0	Enable the accumulation position counter, which is reset by the Index pulse signal with a given phase_A and/or phase_B state. <ul style="list-style-type: none"> 00: Disabled, no position counter reset on the index pulse signal 01: Reset the position counter on the 1st index pulse signal only 10: Reset the position counter on every index pulse signal 11: Reserved
2	RSVD	N/A	-	Reserved
1	POS_RST_PHA	R/W	0	To assign the state of the phase_A signal for the accumulation position counter reset. When CNT_SC is 1, the reset is only valid according to POS_RST_PHA.
0	POS_RST_PHB	R/W	0	To assign the state of the phase_B signal for the accumulation position counter reset. When CNT_SC is 1, this setting is invalid. <p>The accumulation position counter is reset when the index pulse, phase_A and phase_B state matching occur.</p> <ul style="list-style-type: none"> Reset happens when phase_A signal = 0 & phase_B signal = 0. Reset happens when phase_A signal = 0 & phase_B signal = 1. Reset happens when phase_A signal = 1 & phase_B signal = 1. Reset happens when phase_A signal = 1 & phase_B signal = 0.

21.3.3 Velocity Measurement Registers

21.3.3.1 REG_VCTRL

- **Name:** Q-Decoder Velocity Control Register
- **Size:** 32 bits
- **Address offset:** 0x0018
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
RSVD							
23	22	21	20	19	18	17	16
VT_DIV							
R/W							
15	14	13	12	11	10	9	8
RSVD						VMUC_MODE	RSVD
						R/W	
7	6	5	4	3	2	1	0
VUPLMT_INT_EN	VLOWLMT_INT_EN	RSVD		VCCAP_INT_EN	RSVD		VMUC_EN
R/W	R/W			R/W			R/W

Bit	Name	Access	Default	Description
31:24	RSVD	N/A	-	Reserved
23:16	VT_DIV	R/W	0	Divider for the velocity timer clock Velocity Timer Clock = source-clock / (vt_div + 1)
15:10	RSVD	N/A	-	Reserved
9	VMUC_MODE	R/W	0	The velocity measurement unit of measuring the velocity counter mode control <ul style="list-style-type: none"> 0: The counter value uses the absolute value from the decoder 1: The counter value uses the same value of the position counter
8	RSVD	N/A	-	Reserved
7	VUPLMT_INT_EN	R/W	0	Velocity upper limit interrupt enable control <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt When a capture event happens, and the velocity counter capture is bigger than the velocity upper limit, this interrupt is asserted.
6	VLOWLMT_INT_EN	R/W	0	Velocity lower limit interrupt enable control <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt When a capture event happens, and the velocity counter capture is less than the velocity low limit, this interrupt is asserted.
5	RSVD	N/A	-	Reserved
4	VCCAP_INT_EN	R/W	0	Velocity counter capture interrupt enable control <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt The velocity counter register and the position counter are captured in capture registers when the velocity timer reaches zero. This interrupt is asserted when the capture event is detected, and two capture registers have been loaded.
3	RSVD	N/A	-	Reserved
2	VMUC_RST	R/W	0	Reset the velocity measurement unit of measuring the velocity counter <ul style="list-style-type: none"> 0: It is not in reset state. 1: Writing 1 to this bit will reset this unit. Reset the velocity counter and reload the velocity timer. The position counter reset can only be performed when VMUC_EN = 0. Writing 1 to this bit while VMUC_EN = 1 can not reset the position counter.
1	RSVD	N/A	-	Reserved
0	VMUC_EN	R/W	0	The velocity measurement unit of measuring the velocity counter enable control <ul style="list-style-type: none"> 0: Disable 1: Enable

21.3.3.2 REG_VC

- **Name:** Q-Decoder Velocity Counter Register
- **Size:** 32 bits
- **Address offset:** 0x001C
- **Read/write access:** read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VC															
R															

Bit	Name	Access	Default	Description
31:16	RSVD	N/A	-	Reserved
15:0	VC	R	0	The velocity accumulation counter Step is +1, -1 or reset to 0.

				<p>This velocity counter increases or decreases (depending on the direction) by one for every 1 or 2 (depending on CNT_SC) phase states change.</p> <p>For example,</p> <ul style="list-style-type: none"> Case: VMUC_MODE = 0 decode value: +1, +1, +1. vcnt = 3 decode value: +1, -1 , +1. vcnt = 3 decode value: +1, -1 , +1, -1 , -1 , -1 . vcnt = 6 Case: VMUC_MODE = 1 decode value: +1, +1, +1. vcnt = 3 decode value: +1, -1, +1. vcnt = 2 decode value: +1, -1, +1, -1, -1, -1. vcnt = 2
--	--	--	--	--

21.3.3.3 REG_VCCAP

- **Name:** Q-Decoder Velocity Counter Capture Register
- **Size:** 32 bits
- **Address offset:** 0x0020
- **Read/write access:** read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VC_CAP															
R															

Bit	Name	Access	Default	Description
31:16	RSVD	N/A	-	Reserved
15:0	VC_CAP	R	0	When the velocity timer reaches zero, the velocity counter register is captured in the velocity counter capture register.

21.3.3.4 REG_PCCAP

- **Name:** Q-Decoder Position Counter Capture Register
- **Size:** 32 bits
- **Address offset:** 0x0024
- **Read/write access:** read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC_CAP															
R															

Bit	Name	Access	Default	Description
31:16	RSVD	N/A	-	Reserved
15:0	PC_CAP	R	0	When the velocity timer reaches zero, the position counter register is captured in the position counter capture register.

21.3.3.5 REG_VTRLD

- **Name:** Q-Decoder Velocity Time Reload Register
- **Size:** 32 bits
- **Address offset:** 0x0028
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VTRLD															
R/W															

Bit	Name	Access	Default	Description
31:16	RSVD	N/A	-	Reserved
15:0	VTRLD	R/W	0x0	When the velocity timer reaches zero, the velocity timer register reloads this value.

21.3.3.6 REG_VT

- **Name:** Q-Decoder Velocity Timer Register
- **Size:** 32 bits
- **Address offset:** 0x002C
- **Read/write access:** read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT															
R															

Bit	Name	Access	Default	Description
31:16	RSVD	N/A	-	Reserved
15:0	VT	R	0x0	This is the velocity timer value of the down counter.

21.3.3.7 REG_VCOMP

- **Name:** Q-Decoder Velocity Compare Register
- **Size:** 32 bits
- **Address offset:** 0x0030
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VUPLMT															
R/W															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLOWLMT															
R/W															

Bit	Name	Access	Default	Description
31:16	VUPLMT	R/W	0	This is the velocity upper limit value.
15:0	VLOWLMT	R/W	0	This is the velocity lower limit value.

21.3.4 Interrupt Registers

21.3.4.1 REG_IMR

- **Name:** Q-Decoder Interrupt Mask Register
- **Size:** 32 bits
- **Address offset:** 0x003C
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
RSVD							
23	22	21	20	19	18	17	16
RSVD							
15	14	13	12	11	10	9	8
RC_INT_M	RSVD	VUPLMT_INT_M	VLOWLMT_INT_M	RSVD	VCCAP_INT_M	PCE_INT_M	IDX_INT_M
R/W		R/W	R/W		R/W	R/W	R/W
7	6	5	4	3	2	1	0
RUF_INT_M	ROF_INT_M	PC_INT_M	DR_INT_M	IL_INT_M	UF_INT_M	OF_INT_M	CT_INT_M
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Default	Description
31:16	RSVD	N/A	-	Reserved
15	RC_INT_M	R/W	0	Rotation counter comparing interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the rotation counter is equal to the value of 'RCC'.
14	RSVD	N/A	-	Reserved
13	VUPLMT_INT_M	R/W	0	Velocity upper limit interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask When a capture event happens, and the velocity counter capture is bigger than the velocity upper limit, this interrupt is asserted.
12	VLOWLMT_INT_M	R/W	0	Velocity lower limit interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask When a capture event happens, and the velocity counter capture is less than the velocity low limit, this interrupt is asserted.
11	RSVD	N/A	-	Reserved
10	VCCAP_INT_M	R/W	0	Velocity counter capture interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask When the velocity timer reaches zero, the values of velocity counter register and position counter are captured in capture registers. This interrupt is asserted when the capture event is detected, and two capture registers are loaded.
9	PCE_INT_M	R/W	0	Position counter error interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the index pulse signal is detected but the position counter is not equal to 0. This bit is valid only when the index pulse detection is enabled (depending on IDX_EN)
8	IDX_INT_M	R/W	0	Index pulse signal interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the index pulse signal presents. This bit is valid only when the index pulse detection is enabled (depending on IDX_EN)
7	RUF_INT_M	R/W	0	Rotation counter underflow interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the rotation counter underflow occurs (0 → 0xFFFF).
6	ROF_INT_M	R/W	0	Rotation counter overflow interrupt mask

				<ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the rotation counter overflow occurs (0xFF → 0).
5	PC_INT_M	R/W	0	Position counter comparing interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the position counter is equal to the value of 'PCC'.
4	DR_INT_M	R/W	0	Direction changed interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the movement direction changes.
3	IL_INT_M	R/W	0	Illegal state interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the Q-Decoder state error is detected (PHA and PHB change their state concurrently).
2	UF_INT_M	R/W	0	Position counter underflow interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the position counter underflow is occurred.
1	OF_INT_M	R/W	0	Position counter overflow interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the position counter overflow is occurred.
0	CT_INT_M	R/W	0	Counter value changed interrupt mask <ul style="list-style-type: none"> 0: Unmask 1: Mask This interrupt is asserted when the position counter is updated.

21.3.4.2 REG_ISR

- Name:** Q-Decoder Interrupt Status Register
- Size:** 32 bits
- Address offset:** 0x0040
- Read/write access:** read/write

31	30	29	28	27	26	25	24
RSVD							
23	22	21	20	19	18	17	16
RSVD							
15	14	13	12	11	10	9	8
RC_INT_S	RSVD	VUPLMT_INT_S	VLOWLMT_INT_S	RSVD	VCCAP_INT_S	PCE_INT_S	IDX_INT_S
R/W		R/W	R/W		R/W	R/W	R/W
7	6	5	4	3	2	1	0
RUF_INT_S	ROF_INT_S	PC_INT_S	DR_INT_S	IL_INT_S	UF_INT_S	OF_INT_S	CT_INT_S
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Default	Description
31:16	RSVD	N/A	-	Reserved
15	RC_INT_S	R/WC	0	<ul style="list-style-type: none"> 0: No interrupt 1: Interrupt is pending
14	RSVD	N/A	-	<ul style="list-style-type: none"> Reserved
13	VUPLMT_INT_S	R/WC	0	<ul style="list-style-type: none"> 0: No interrupt 1: Interrupt is pending

12	VLOWLMT_INT_S	R/WC	0	<ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending
11	RSVD	N/A	-	<ul style="list-style-type: none"> ● Reserved
10	VCCAP_INT_S	R/WC	0	<ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending
9	PCE_INT_S	R/WC	0	Position counter error interrupt status <ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending This bit is valid only when the index pulse detection is enabled (depending on IDX_EN). Writing 1 to this bit clears this interrupt status.
8	IDX_INT_S	R/WC	0	Index pulse signal interrupt status <ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending This bit is valid only when the index pulse detection is enabled (depending on IDX_EN). Writing 1 to this bit clears this interrupt status.
7	RUF_INT_S	R/WC	0	Rotation counter underflow interrupt status <ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending Writing 1 to this bit clears this interrupt status.
6	ROF_INT_S	R/WC	0	Rotation counter overflow interrupt status <ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending Writing 1 to this bit clears this interrupt status.
5	PC_INT_S	R/WC	0	Position counter comparing interrupt status <ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending Writing 1 to this bit clears this interrupt status.
4	DR_INT_S	R/WC	0	Direction changed interrupt status <ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending Writing 1 to this bit clears this interrupt status.
3	IL_INT_S	R/WC	0	Illegal state interrupt status <ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending Writing 1 to this bit clears this interrupt status.
2	UF_INT_S	R/WC	0	Position counter underflow interrupt status <ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending Writing 1 to this bit clears this interrupt status.
1	OF_INT_S	R/WC	0	Position counter overflow interrupt status <ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending Writing 1 to this bit clears this interrupt status.
0	CT_INT_S	R/WC	0	Position counter changed interrupt status <ul style="list-style-type: none"> ● 0: No interrupt ● 1: Interrupt is pending Writing 1 to this bit clears this interrupt status.

22 Inter-IC Sound (I²S)

22.1 Introduction

Inter-IC Sound (I²S) is an electrical serial bus interface standard used for communication with digital audio devices. It communicates audio data between integrated circuits in an electronic device. The I²S bus separates clock and serial data signals, resulting in a lower jitter than typical communication systems that recover the clock from the data stream. The I²S is widely used in various multimedia systems.

The I²S can operate in slave or master mode, as a receiver or a transmitter. It can address three different audio standards including the I²S Philips standard, the left-justified standard and right-justified standard.

22.2 Features

- Sample bit: 16-bit, 24-bit, 32-bit
- Sample rate: 8k, 12k, 16k, 24k, 32k, 48k, 64k, 96k, 192k, 384k, 7.35k, 11.025k, 14.7k, 22.05k, 29.4k, 44.1k, 58.8k, 88.2k, 176.4k
- I²S throughput: 0.2352Mbps (7.35k * 32bit) ~ 24.576Mbps (384k * 64bit)
- I²S channel number: mono, stereo, 5.1 channel
- Sample bit for mono: 16-bit, 32-bit
- Sample bit for stereo & 5.1 channel: 16-bit, 24-bit, 32-bit
- Integrated DMA engine to minimize the software efforts
- Master or slave mode
- Mono and stereo Tx or Rx, or Tx & Rx mode
- 5.1 Tx mode (DAC) supported only, Rx mode (ADC) not supported
- Not for PCM mode
- Mute function

22.3 Interface

The external interface of I²S is I²S out, which contains Master Clock (MCK), Serial Clock (SCK), Word Select (WS) and Serial Data (SD). As for I²S out, according to different channel number of audio data, the configuration varies. For mono and stereo audio, SD regards SD_i as input and SD_o as output. Fig 1-1 shows I²S mono/stereo audio-out interface configuration. For 5.1 channel audio, because Ameba-D only supports 5.1 Tx mode, SD contains SD0, SD1, SD2 as output without input. SD0 transmits audio data in the left channel(A) and the right channel(A), SD1 in the left channel(B) and the right channel(B), and SD2 in the left channel(C) and the right channel(C). Fig 1-2 shows I²S 5.1 channel audio-out interface configuration.

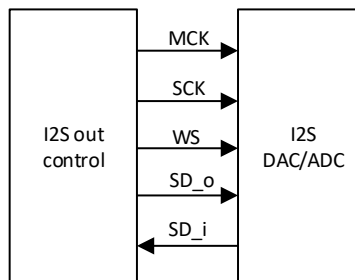


Fig 22-1 I²S mono/stereo audio-out interface configuration

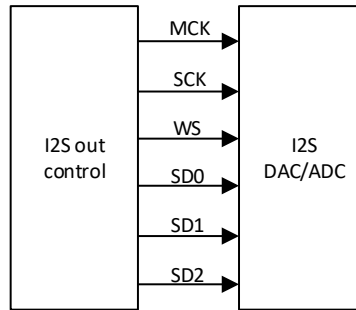


Fig 22-2 I2S 5.1 channel audio-out interface configuration

22.4 Functional Description

22.4.1 Signal Lines

Signal lines in I2S data format are shown in Fig 1-3.

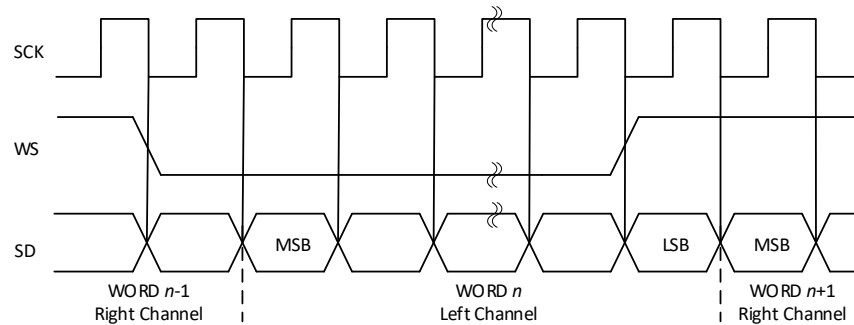


Fig 22-3 Signal lines in I2S data format

22.4.1.1 Serial Clock

- Serial clock is a synchronous bit clock. Each data bit has a pulse.
- Serial clock is generated by internal in master mode, and provided by external in slave mode.
- $SCK\ rate = 2 * sample\ rate * sample\ bit$

22.4.1.2 Word Select

- Word Select (WS) indicates the channel being transmitted.
 - WS = 0: Channel 1 (left)
 - WS = 1: Channel 2 (right)
- There is no need to be symmetrical, WS changes on either a trailing or a leading edge of SCK.
- In I2S data format, WS changes one clock period before the MSB is transmitted.
 - Allow the slave transmitter to derive synchronous timing of the SD.
 - Enable the receiver to store the previous data word and clear the input for the next data word.
- WS changes during a WS period, so the left channel data is transmitted when WS stays low and the right channel data is transmitted when WS stays high.
- In mono mode, WS changes the same as in stereo mode, the difference is that there is no data transmitted when WS=1 (the right channel).
- Frequency of WS equals to the sample rate.

22.4.1.3 Serial Data

- Serial Data (SD) line is designed into two lines (SD_i, SD_o) for Ameba-D I²S except in 5.1 Tx mode, in which three lines (SD0, SD1, SD2) are used.
- SD is transmitted with MSB first.
- In I²S data format, MSB has a fixed position, while the position of the LSB depends on the word length.
- When system word length is greater than transmitter word length, the word is truncated (least significant data bits are set to '0').
 - If receiver is sent more bits than its word length, the bits after the LSB are ignored.
 - If receiver is sent fewer bits than its word length, the missing bits are set to '0' internally.
- In I²S data format, transmitter always sends MSB of the next word one clock period after the WS changes.
- SD sent by transmitter is synchronized with either the trailing (HIGH to LOW) or the leading (LOW to HIGH) edge of the clock signal.
- SD must be latched into the receiver on the leading edge of SCK.

22.4.2 Operation Mode

22.4.2.1 Transmitter as the Master

As Fig 22-4 shows, as the master in this operation mode, the transmitter has to generate the SCK, WS signal and SD, while the receiver accepts the SCK, WS, SD from the bus as the slave.

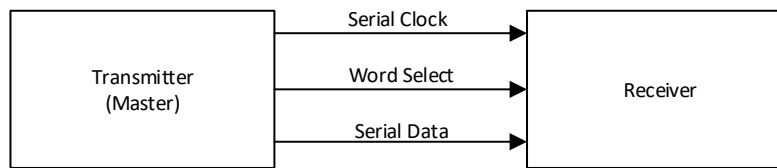


Fig 22-4 Transmitter as the master

22.4.2.2 Receiver as the Master

As Fig 22-5 shows, as the master in this operation mode, the receiver has to generate the SCK and WS signal, while the transmitter sends SD under the control of SCK and WS as the slave.

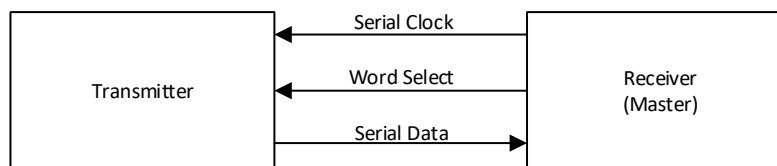


Fig 22-5 Receiver as the master

22.4.2.3 Controller as the Master

As Fig 22-6 shows, as the master in this operation mode, the controller generates the SCK and WS signal, while the transmitter and the receiver act as the slave. The transmitter has to generate data under the control of external clocks.

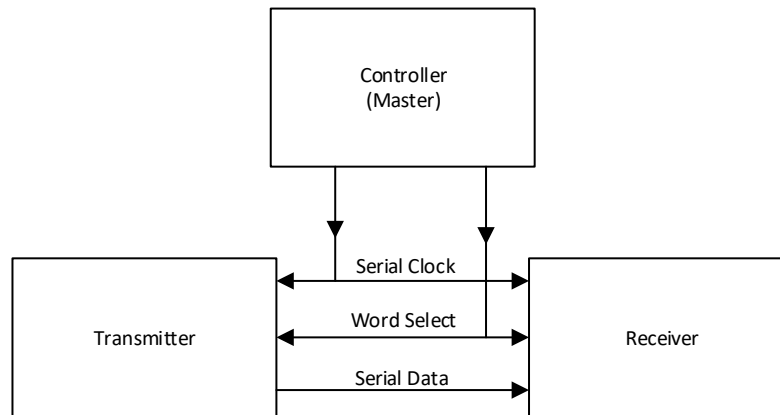


Fig 22-6 Controller as the master

22.4.3 Serial Data Standard

22.4.3.1 I²S Philips Standard

For the I²S Philips standard, you should focus on the following items:

- MSB is sent one SCK cycle after WS changes, as Fig 22-7 shows.
- For stereo, the left channel is transmitted when WS = 0 and the right channel is transmitted when WS = 1.
- The transmitter and the receiver may have different word length.

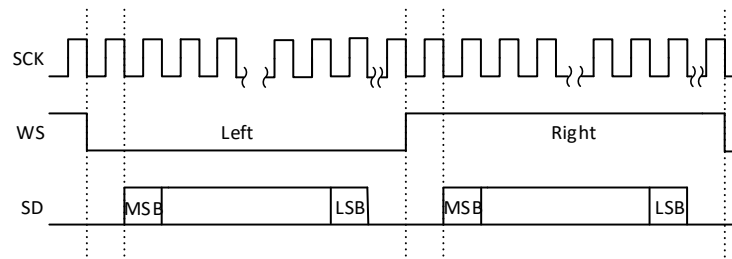


Fig 22-7 I²S Philips standard

22.4.3.2 Left-justified Standard

The left-justified standard is rarely used. For this standard, you should focus on the following items:

- Serial data is left justified, as Fig 22-8 shows.
- For stereo, the left channel is transmitted when WS = 1 and the right channel is transmitted when WS = 0.
- The transmitter and the receiver must have the same word length.

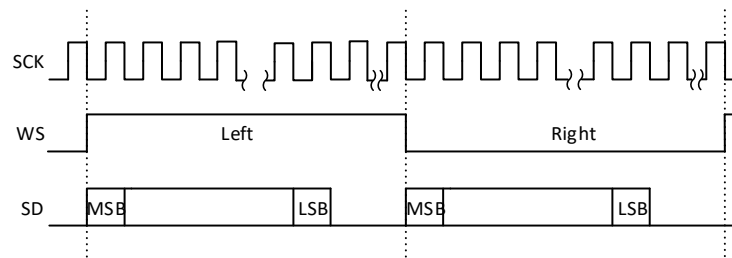


Fig 22-8 Left-justified standard

22.4.3.3 Right-justified Standard

For the right-justified standard, you should focus on the following items:

- Serial data is right justified, as Fig 22-9 shows.
- For stereo, the left channel is transmitted when WS = 1 and the right channel is transmitted when WS = 0.
- The transmitter and the receiver must have the same word length.

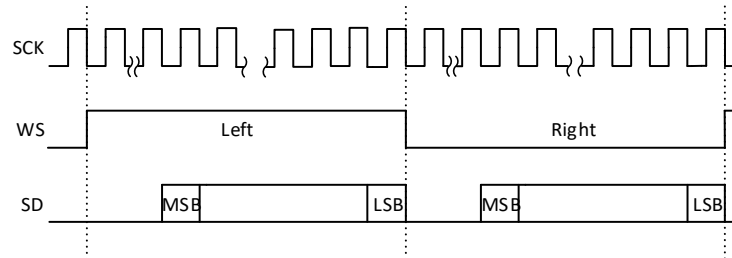


Fig 22-9 Right-justified standard

22.4.4 Clock Type

The I²S clock tree is shown in Fig 1-10.

- The source clock is 98.304MHz and 45.1584MHz.
- MCK, SCK and WS signals are divided by the source clock.

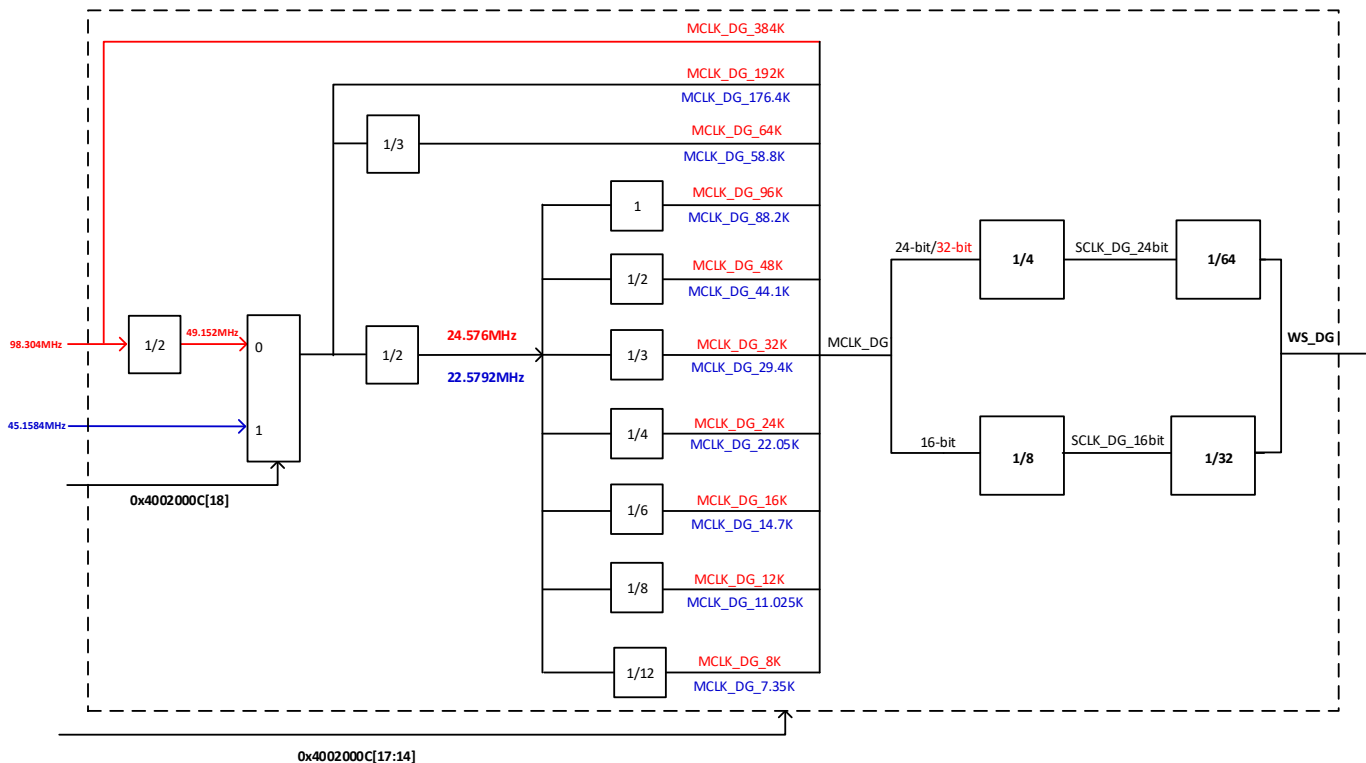


Fig 22-10 I²S clock tree

The relationship between MCK, WS and SCK is as follows:

- If sample bit is 16-bit, MCK=8SCK=256WS.
- If sample bit is 24-bit, MCK=4SCK=256WS.

- If sample bit is 32-bit, $MCK=4SCK=256WS$.

The frequency of WS is equal to the sample rate. The types of sample rate supported by Ameba-D are 384kHz, 192kHz, 96kHz, 64kHz, 48kHz, 32kHz, 24kHz, 16kHz, 12kHz, 8kHz ($\pm 50PPM$) and 176.4kHz, 88.2kHz, 58.8kHz, 44.1kHz, 29.4kHz, 22.05kHz, 14.7kHz, 11.025kHz, 7.35kHz ($\pm 50PPM$). Table 1-1 shows the details of clock configuration.

Table 22-1 Clock configuration

Sample Rate	WS	SCK (16-bit)	SCK (24-bit/32-bit)	MCK
384kHz	384kHz	12.288MHz	24.576MHz	98.304MHz
192kHz	192kHz	6.144MHz	12.288MHz	49.152MHz
96kHz	96kHz	3.072MHz	6.144MHz	24.576MHz
64kHz	64kHz	2.048MHz	4.096MHz	16.384MHz
48kHz	48kHz	1.536MHz	3.072MHz	12.288MHz
32kHz	32kHz	1.024MHz	2.048MHz	8.192MHz
24kHz	24kHz	0.768MHz	1.536MHz	6.144MHz
16kHz	16kHz	0.512MHz	1.024MHz	4.096MHz
12kHz	12kHz	0.384MHz	0.768MHz	3.072MHz
8kHz	8kHz	0.256MHz	0.512MHz	2.048MHz
176.4kHz	176.4kHz	5.6448MHz	11.2896MHz	45.1584MHz
88.2kHz	88.2kHz	2.8224MHz	5.6448MHz	22.5792MHz
58.8kHz	58.8kHz	1.8816MHz	3.7632MHz	15.0528MHz
44.1kHz	44.1kHz	1.4112MHz	2.8224MHz	11.2896MHz
29.4kHz	29.4kHz	0.9408MHz	1.8816MHz	7.5264MHz
22.05kHz	22.05kHz	0.7056MHz	1.4112MHz	5.6448MHz
14.7kHz	14.7kHz	0.4704MHz	0.9408MHz	3.7632MHz
11.025kHz	11.025kHz	0.3528MHz	0.7056MHz	2.8224MHz
7.35kHz	7.35kHz	0.2352MHz	0.4704MHz	1.8816MHz

22.4.5 Memory Block

As for memory block, page pointer, page size and page number are set by writing I²S registers like Fig 22-11. The max. page number is 4 and the max. page size is 16K bytes. The owner of every memory page can be set to CPU or I²S controller by changing page own bit in I²S registers.

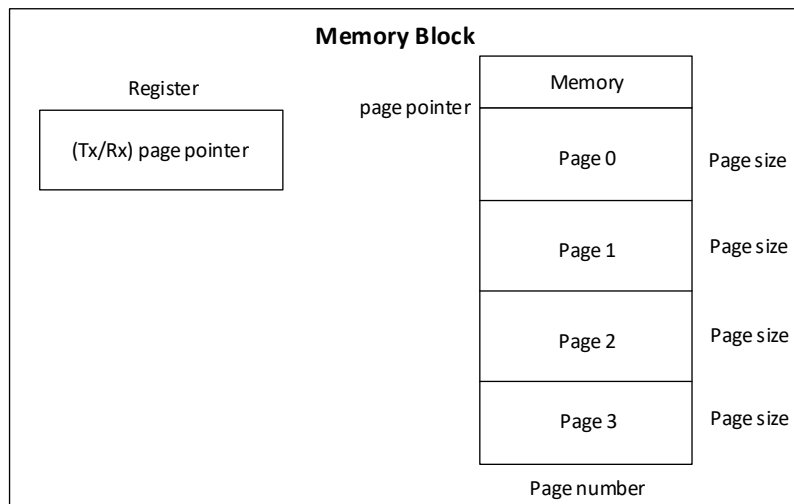


Fig 22-11 Memory block

22.4.6 FIFO Allocation

- Asynchronous FIFO is used to buffer data from memory to I²S controller.
- I²S informs channel numbers and sample size, so data can be exactly extracted from the FIFO.
- FIFO size is 64*32 bits (256 bytes), FIFO depth is 64, and burst length is set by writing I²S registers.
- Data is filled in the unit of row.

22.4.6.1 Mono Channel (FIFO)

When the sample bit is 16-bit or 32-bit, the FIFO allocation of the mono channel is illustrated in Fig 22-12 and Fig 22-13.

Sample bit = 16-bit	
31 MSB	0 LSB
Mono channel(1)	Mono channel(2)
Mono channel(3)	Mono channel(4)
Mono channel(5)	Mono channel(6)
Mono channel(7)	Mono channel(8)
Mono channel(9)	Mono channel(10)
Mono channel(11)	Mono channel(12)
Mono channel(13)	Mono channel(14)
Mono channel(15)	Mono channel(16)

Fig 22-12 FIFO allocation of mono channel (sample bit = 16-bit)

Sample bit = 32-bit	
31 MSB	0 LSB
Mono channel(1)	
Mono channel(2)	
Mono channel(3)	
Mono channel(4)	
Mono channel(5)	
Mono channel(6)	
Mono channel(7)	
Mono channel(8)	

Fig 22-13 FIFO allocation of mono channel (sample bit = 32-bit)

22.4.6.2 Stereo Channel (FIFO)

When the sample bit is 16-bit, 24-bit or 32-bit, the FIFO allocation of the stereo channel is illustrated in Fig 22-14 to Fig 22-16.

31	Sample bit = 16-bit		0
MSB	Left channel	Right channel	LSB
	Left channel	Right channel	
	Left channel	Right channel	
	Left channel	Right channel	
	Left channel	Right channel	
	Left channel	Right channel	
	Left channel	Right channel	
	Left channel	Right channel	
	Left channel	Right channel	

Fig 22-14 FIFO allocation of stereo channel (sample bit = 16-bit)

31	Sample bit = 24-bit		0
MSB	8'b0	Left channel	LSB
	8'b0	Right channel	
	8'b0	Left channel	
	8'b0	Right channel	
	8'b0	Left channel	
	8'b0	Right channel	
	8'b0	Left channel	
	8'b0	Right channel	

Fig 22-15 FIFO allocation of stereo channel (sample bit = 24-bit)

31	Sample bit = 32-bit		0
MSB	Left channel		LSB
	Right channel		
	Left channel		
	Right channel		
	Left channel		
	Right channel		
	Left channel		
	Right channel		

Fig 22-16 FIFO allocation of stereo channel (sample bit = 32-bit)

22.4.6.3 5.1 Channel (FIFO)

When the sample bit is 16-bit, 24-bit or 32-bit, the FIFO allocation of the 5.1 channel is illustrated in Fig 22-17 to Fig 22-19.

Sample bit = 16-bit	
31 MSB	0 LSB
Left channel(A)	Right channel(A)
Left channel(B)	Right channel(B)
Left channel(C)	Right channel(C)
Left channel(A)	Right channel(A)
Left channel(B)	Right channel(B)
Left channel(C)	Right channel(C)
Left channel(A)	Right channel(A)
Left channel(B)	Right channel(B)

Fig 22-17 FIFO allocation of 5.1 channel (sample bit = 16-bit)

Sample bit = 24-bit	
31 MSB	0 LSB
8'b0	Left channel(A)
8'b0	Left channel(B)
8'b0	Left channel(C)
8'b0	Right channel(A)
8'b0	Right channel(B)
8'b0	Right channel(C)
8'b0	Left channel(A)
8'b0	Left channel(B)

Fig 22-18 FIFO allocation of 5.1 channel (sample bit = 24-bit)

Sample bit = 32-bit	
31 MSB	0 LSB
Left channel(A)	
Left channel(B)	
Left channel(C)	
Right channel(A)	
Right channel(B)	
Right channel(C)	
Left channel(A)	
Left channel(B)	

Fig 22-19 FIFO allocation of 5.1 channel (sample bit = 32-bit)

22.5 Registers

The details of the I²S memory map are listed in Table 22-2. The base address of I²S is 0x4002_0000, and the total size is 1KB.

Table 22-2 Memory map of I²S

Name	Address Offset	Access	Description
IS_CTL	0x0000	R/W	I ² S Control Register
IS_TX_PAGE_PTR	0x0004	R/W	I ² S Tx Page Pointer Register
IS_RX_PAGE_PTR	0x0008	R/W	I ² S Rx Page Pointer Register
IS_SETTING	0x000C	R/W	I ² S Page Size and Sample Rate Setting Register
IS_TX_MASK_INT	0x0010	R/W	I ² S Tx Interrupt Enable Register
IS_TX_STATUS_INT	0x0014	R/W	I ² S Tx Interrupt Status Register
IS_RX_MASK_INT	0x0018	R/W	I ² S Rx Interrupt Enable Register
IS_RX_STATUS_INT	0x001C	R/W	I ² S Rx Interrupt Status Register
IS_TX_PAGE_OWN0	0x0020	R/W	I ² S Tx Page 0 Own Bit Register
IS_TX_PAGE_OWN1	0x0024	R/W	I ² S Tx Page 1 Own Bit Register
IS_TX_PAGE_OWN2	0x0028	R/W	I ² S Tx Page 2 Own Bit Register
IS_TX_PAGE_OWN3	0x002C	R/W	I ² S Tx Page 3 Own Bit Register
IS_RX_PAGE_OWN0	0x0030	R/W	I ² S Rx Page 0 Own Bit Register
IS_RX_PAGE_OWN1	0x0034	R/W	I ² S Rx Page 1 Own Bit Register
IS_RX_PAGE_OWN2	0x0038	R/W	I ² S Rx Page 2 Own Bit Register
IS_RX_PAGE_OWN3	0x003C	R/W	I ² S Rx Page 3 Own Bit Register
IS_VERSION_ID	0x0040	R	I ² S Version ID Register

22.5.1 Control Register (IS_CTL)

- **Name:** I²S Control Register
- **Size:** 32 bits
- **Address offset:** 0x0000
- **Read/write access:** read/write

31	30	29	28	27	26	25	24
SW_RST	WL	SLAVE_MODE	MUTE	RSVD			
R/W	R/W	R/W	R/W				
23	22	21	20	19	18	17	16
RSVD	BURST_SIZE					RSVD	DEBUG_SWITCH
	R/W						R/W
15	14	13	12	11	10	9	8
DEBUG_SWITCH	RSVD		BYTE_SWAP	SCK_SWAP	DACLRSWAP	FORMAT	
R/W			R/W	R/W	R/W	R/W	
7	6	5	4	3	2	1	0
LOOP_BACK	RSVD	EDGE_SW	AUDIO_MONO		TX_ACT		I2S_EN
R/W		R/W	R/W		R/W		R/W

Bit	Name	Access	Reset	Description
31	SW_RST	R/W	0x0	<ul style="list-style-type: none"> ● 0: Software reset ● 1: No software reset
30:29	WL	R/W	0x0	Word length <ul style="list-style-type: none"> ● 00: 16 bits ● 01: 24 bits ● 10: 32 bits ● 11: Unused (error)
28	SLAVE_MODE	R/W	0x0	Slave mode <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable
27	MUTE	R/W	0x0	Mute function

				<ul style="list-style-type: none"> 0: Disable 1: Enable
26:23	RSVD	N/A	-	Reserved
22:18	BURST_SIZE	R/W	0xf	<ul style="list-style-type: none"> 5'h0_1111: Burst16 5'h0_1011: Burst12 5'h0_0111: Burst8 5'h0_0011: Burst4
17	RSVD	N/A	-	Reserved
16:15	DEBUG_SWITCH	R/W	0x0	Switch 32-bit debug signal
14:13	RSVD	N/A	-	Reserved
12	BYTE_SWAP	R/W	0x0	Byte swap for big-endian or little-endian <ul style="list-style-type: none"> 0: Enable for big-endian 1: Enable for little-endian
11	SCK_SWAP	R/W	0x0	Invert serial clock <ul style="list-style-type: none"> 0: Disable 1: Enable
10	DACLRSWAP	R/W	0x0	Control whether DAC appears in right phase or left phase of WS clock <ul style="list-style-type: none"> 0: Left phase 1: Right phase
9:8	FORMAT	R/W	0x0	Digital Interface Format <ul style="list-style-type: none"> 00: I²S 01: Left-justified 10: Right-justified
7	LOOP_BACK	R/W	0x0	Internal testing
6	RSVD	N/A	-	Reserved
5	EDGE_SW	R/W	0x0	Edge switch <ul style="list-style-type: none"> 0: Negative edge 1: Positive edge
4:3	AUDIO_MONO	R/W	0x0	Audio mono <ul style="list-style-type: none"> 00: Stereo audio 01: 5.1 Audio 10: Mono
2:1	TX_ACT	R/W	0x0	Tx activity <ul style="list-style-type: none"> 00: Rx path 01: Tx path (internal LBK need to be enabled) 10: Tx_Rx path (Bi-direction audio/voice application, not involving 5.1 audio)
0	I2S_EN	R/W	0x0	I ² S enable <ul style="list-style-type: none"> 0: Disable 1: Enable

22.5.2 Tx Page Pointer Register (IS_TX_PAGE_PTR)

- Name:** I²S Tx Page Pointer Register
- Size:** 32 bits
- Address offset:** 0x0004
- Read/write access:** read/write

31	30	29	...	4	3	2	1	0
PAGE_PTR_TX								RSVD
R/W								

Bit	Name	Access	Reset	Description
31:2	PAGE_PTR_TX	R/W	0x0	Tx page pointer. This is a physical address with word-aligned limitation.
1:0	RSVD	N/A	-	Reserved

22.5.3 Rx Page Pointer Register (IS_RX_PAGE_PTR)

- **Name:** I²S Rx Page Pointer Register
- **Size:** 32 bits
- **Address offset:** 0x0008
- **Read/write access:** read/write

31	30	29	...	4	3	2	1	0
PAGE_PTR_RX								RSVD
R/W								

Bit	Name	Access	Reset	Description
31:2	PAGE_PTR_RX	R/W	0x0	Rx page pointer. This is a physical address with word-aligned limitation.
1:0	RSVD	N/A	-	Reserved

22.5.4 Page Size and Sample Rate Setting Register (IS_SETTING)

- **Name:** I²S Page Size and Sample Rate Setting Register
- **Size:** 32 bits
- **Address offset:** 0x000C
- **Read/write access:** read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD													CLK_SWITCH	SR	
													R/W	R/W	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SR		PAGE_NUM		PAGE_SIZE											
R/W		R/W		R/W											

Bit	Name	Access	Reset	Description
31:19	RSVD	N/A	-	Reserved
18	CLK_SWITCH	R/W	0x0	I ² S clock switch <ul style="list-style-type: none"> ● 0: CLK_I2S = 98.304MHz (±50ppm) ● 1: CLK_I2S = 45.1584MHz (±50ppm)
17:14	SR	R/W	0101	Sample rate <ul style="list-style-type: none"> ● 0000: 8kHz/7.35kHz ● 0001: 12kHz/11.025kHz ● 0010: 16kHz/14.7kHz ● 0011: 24kHz/22.05kHz ● 0100: 32kHz/29.4kHz ● 0101: 48kHz/44.1kHz ● 0110: 64kHz/58.8kHz ● 0111: 96kHz/88.2kHz ● 1000: 192kHz/176.4kHz ● 1001: 384kHz ● Others: Reserved
13:12	PAGE_NUM	R/W	0x0	Page number
11:0	PAGE_SIZE	R/W	0x0	Page size (word unit)

22.5.5 Tx Interrupt Enable Register (IS_TX_MASK_INT)

- **Name:** I²S Tx Interrupt Enable Register
- **Size:** 32 bits
- **Address offset:** 0x0010
- **Read/write access:** read/write

31	30	29	28	...	12	11	10	9
RSVD								
8	7	6	5	4	3	2	1	0
FIFO_EMPTY_I E_TX	PAGEUNAVA_I E_TX3	PAGEUNAVA_I E_TX2	PAGEUNAVA_I E_TX1	PAGEUNAVA_I E_TX0	P3OKIE_TX	P2OKIE_TX	P1OKIE_TX	P0OKIE_TX
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:9	RSVD	N/A	-	Reserved
8	FIFO_EMPTY_IE_TX	R/W	0x0	Tx FIFO Empty Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
7	PAGEUNAVA_IE_TX3	R/W	0x0	Tx Page 3 Unavailable Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
6	PAGEUNAVA_IE_TX2	R/W	0x0	Tx Page 2 Unavailable Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
5	PAGEUNAVA_IE_TX1	R/W	0x0	Tx Page 1 Unavailable Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
4	PAGEUNAVA_IE_TX0	R/W	0x0	Tx Page 0 Unavailable Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
3	P3OKIE_TX	R/W	0x0	Tx Page 3 OK Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
2	P2OKIE_TX	R/W	0x0	Tx Page 2 OK Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
1	P1OKIE_TX	R/W	0x0	Tx Page 1 OK Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
0	P0OKIE_TX	R/W	0x0	Tx Page 0 OK Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt

22.5.6 Tx Interrupt Status Register (IS_TX_STATUS_INT)

- **Name:** I²S Tx Interrupt Status Register
- **Size:** 32 bits
- **Address offset:** 0x0014
- **Read/write access:** read/write

31	30	29	28	...	12	11	10	9
RSVD								
8	7	6	5	4	3	2	1	0
FIFO_EMPTY_I P_TX	PAGEUNAVA_I P_TX3	PAGEUNAVA_I P_TX2	PAGEUNAVA_I P_TX1	PAGEUNAVA_I P_TX0	P3OKIP_TX	P2OKIP_TX	P1OKIP_TX	P0OKIP_TX
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:9	RSVD	N/A	-	Reserved
8	FIFO_EMPTY_IP_TX	R/W	0x0	Tx FIFO Empty Interrupt Pending <ul style="list-style-type: none"> 0: No Interrupt

				<ul style="list-style-type: none"> 1: Interrupt pending, writing '1' to clear
7	PAGEUNAVA_IP_TX3	R/W	0x0	Tx Page 3 Unavailable Interrupt Pending <ul style="list-style-type: none"> 0: No Interrupt 1: Interrupt pending, writing '1' to clear
6	PAGEUNAVA_IP_TX2	R/W	0x0	Tx Page 2 Unavailable Interrupt Pending <ul style="list-style-type: none"> 0: No Interrupt 1: Interrupt pending, writing '1' to clear
5	PAGEUNAVA_IP_TX1	R/W	0x0	Tx Page 1 Unavailable Interrupt Pending <ul style="list-style-type: none"> 0: No Interrupt 1: Interrupt pending, writing '1' to clear
4	PAGEUNAVA_IP_TX0	R/W	0x0	Tx Page 0 Unavailable Interrupt Pending <ul style="list-style-type: none"> 0: No Interrupt 1: Interrupt pending, writing '1' to clear
3	P3OKIP_TX	R/W	0x0	Tx Page 3 OK Interrupt Pending <ul style="list-style-type: none"> 0: No Interrupt 1: Interrupt pending, writing '1' to clear
2	P2OKIP_TX	R/W	0x0	Tx Page 2 OK Interrupt Pending <ul style="list-style-type: none"> 0: No Interrupt 1: Interrupt pending, writing '1' to clear
1	P1OKIP_TX	R/W	0x0	Tx Page 1 OK Interrupt Pending <ul style="list-style-type: none"> 0: No Interrupt 1: Interrupt pending, writing '1' to clear
0	P0OKIP_TX	R/W	0x0	Tx Page 0 OK Interrupt Pending <ul style="list-style-type: none"> 0: No Interrupt 1: Interrupt pending, writing '1' to clear

22.5.7 Rx Interrupt Enable Register (IS_RX_MASK_INT)

- Name:** I²S Rx Interrupt Enable Register
- Size:** 32 bits
- Address offset:** 0x0018
- Read/write access:** read/write

31	30	29	28	...	12	11	10	9
RSVD								
8	7	6	5	4	3	2	1	0
FIFO_FULL_IE_RX	PAGEUNAVA_I_E_RX3	PAGEUNAVA_I_E_RX2	PAGEUNAVA_I_E_RX1	PAGEUNAVA_I_E_RX0	P3OKIE_RX	P2OKIE_RX	P1OKIE_RX	P0OKIE_RX
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:9	RSVD	N/A	-	Reserved
8	FIFO_FULL_IE_RX	R/W	0x0	Rx FIFO Full Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
7	PAGEUNAVA_IE_RX3	R/W	0x0	Rx Page 3 Unavailable Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
6	PAGEUNAVA_IE_RX2	R/W	0x0	Rx Page 2 Unavailable Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
5	PAGEUNAVA_IE_RX1	R/W	0x0	Rx Page 1 Unavailable Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
4	PAGEUNAVA_IE_RX0	R/W	0x0	Rx Page 0 Unavailable Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt

				<ul style="list-style-type: none"> 1: Enable interrupt
3	P3OKIE_RX	R/W	0x0	Rx Page 3 OK Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
2	P2OKIE_RX	R/W	0x0	Rx Page 2 OK Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
1	P1OKIE_RX	R/W	0x0	Rx Page 1 OK Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt
0	P0OKIE_RX	R/W	0x0	Rx Page 0 OK Interrupt Enable <ul style="list-style-type: none"> 0: Disable interrupt 1: Enable interrupt

22.5.8 Rx Interrupt Status Register (IS_RX_STATUS_INT)

- Name:** I²S Rx Interrupt Status Register
- Size:** 32 bits
- Address offset:** 0x001C
- Read/write access:** read/write

31	30	29	28	...	12	11	10	9
RSVD								
8	7	6	5	4	3	2	1	0
FIFO_FULL_IP_RX	PAGEUNAVA_IP_RX3	PAGEUNAVA_IP_RX2	PAGEUNAVA_IP_RX1	PAGEUNAVA_IP_RX0	P3OKIP_RX	P2OKIP_RX	P1OKIP_RX	P0OKIP_RX
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Access	Reset	Description
31:9	RSVD	N/A	-	Reserved
8	FIFO_FULL_IP_RX	R/W	0x0	Rx FIFO Full Interrupt Pending <ul style="list-style-type: none"> 0: No interrupt 1: Interrupt pending, writing '1' to clear
7	PAGEUNAVA_IP_RX3	R/W	0x0	Rx Page 3 Unavailable Interrupt Pending <ul style="list-style-type: none"> 0: No interrupt 1: Interrupt pending, writing '1' to clear
6	PAGEUNAVA_IP_RX2	R/W	0x0	Rx Page 2 Unavailable Interrupt Pending <ul style="list-style-type: none"> 0: No interrupt 1: Interrupt pending, writing '1' to clear
5	PAGEUNAVA_IP_RX1	R/W	0x0	Rx Page 1 Unavailable Interrupt Pending <ul style="list-style-type: none"> 0: No interrupt 1: Interrupt pending, writing '1' to clear
4	PAGEUNAVA_IP_RX0	R/W	0x0	Rx Page 0 Unavailable Interrupt Pending <ul style="list-style-type: none"> 0: No interrupt 1: Interrupt pending, writing '1' to clear
3	P3OKIP_RX	R/W	0x0	Rx Page 3 OK Interrupt Pending <ul style="list-style-type: none"> 0: No interrupt 1: Interrupt pending, writing '1' to clear
2	P2OKIP_RX	R/W	0x0	Rx Page 2 OK Interrupt Pending <ul style="list-style-type: none"> 0: No interrupt 1: Interrupt pending, writing '1' to clear
1	P1OKIP_RX	R/W	0x0	Rx Page 1 OK Interrupt Pending <ul style="list-style-type: none"> 0: No interrupt 1: Interrupt pending, writing '1' to clear
0	P0OKIP_RX	R/W	0x0	Rx Page 0 OK Interrupt Pending <ul style="list-style-type: none"> 0: No interrupt

				● 1: Interrupt pending, writing '1' to clear
--	--	--	--	--

22.5.9 Tx Page Own Bit Register (IS_TX_PAGE_OWNNx)

- **Name:** I²S Tx Page x Own Bit Register (x = 0, 1, 2, 3)
- **Size:** 32 bits
- **Address offset:** for x = 0 to 3:
 - IS_TX_PAGE_OWNO: 0x0020
 - IS_TX_PAGE_OWNN1: 0x0024
 - IS_TX_PAGE_OWNN2: 0x0028
 - IS_TX_PAGE_OWNN3: 0x002C
- **Read/write access:** read/write

31	30	29	28	27	...	3	2	1	0
TX_PAGEOWNx	RSVD								
R/W									

Bit	Name	Access	Reset	Description
31	TX_PAGEOWNx	R/W	0x0	Tx Page x (x = 0, 1, 2, 3) own bit <ul style="list-style-type: none"> ● 0: Page x owned by CPU ● 1: Page x owned by I²S controller
30:0	RSVD	N/A	-	Reserved

22.5.10 Rx Page Own Bit Register (IS_RX_PAGE_OWNNx)

- **Name:** I²S Rx Page x Own Bit Register (x = 0, 1, 2, 3)
- **Size:** 32 bits
- **Address offset:** for x = 0 to 3:
 - IS_RX_PAGE_OWNO: 0x0030
 - IS_RX_PAGE_OWNN1: 0x0034
 - IS_RX_PAGE_OWNN2: 0x0038
 - IS_RX_PAGE_OWNN3: 0x003C
- **Read/write access:** read/write

31	30	29	28	27	...	3	2	1	0
RX_PAGEOWNx	RSVD								
R/W									

Bit	Name	Access	Reset	Description
31	RX_PAGEOWNx	R/W	0x0	Rx Page x (x = 0, 1, 2, 3) own bit <ul style="list-style-type: none"> ● 0: Page x owned by CPU ● 1: Page x owned by I²S controller
30:0	RSVD	N/A	-	Reserved

22.5.11 Version ID (IS_VERSION_ID)

- **Name:** I²S Version ID Register
- **Size:** 32 bits
- **Address offset:** 0x0040
- **Read/write access:** read-only

31	30	29	28	...	3	2	1	0
VERSION_ID								
R								

Bit	Name	Access	Reset	Description
31:0	VERSION_ID	R	0x2016_0001	I ² S version ID

Abbreviations

ACC	Audio Codec Controller
ADC	Analog-to-Digital Converter
AMBA	Advanced Microcontroller Bus Architecture — a trademarked name by ARM Limited that defines an on-chip communication standard for high speed microcontrollers.
AON	Always on
APB	Advanced Peripheral Bus — optimized for minimal power consumption and reduced interface complexity to support peripheral functions (ARM Limited specification).
arbiter	AMBA bus submodule that arbitrates bus activity between masters and slaves.
AXI	Advanced eXtensible Interface — a high-performance on-chip bus defined by ARM.
BFM	Bus-Functional Model — A simulation model used for early hardware debug. A BFM simulates the bus cycles of a device and models device pins, as well as certain on-chip functions. See also Full-Functional Model.
BLE	Bluetooth Low Energy, a wireless local area network technology.
BT	Bluetooth
BTN	Button
bus bridge	Logic that handles the interface and transactions between two bus standards, such as AHB and APB.
CA	Command Address
CAP	Capacitive
CEN	Counter enable
C _F	Capacitance of Finger
C _P	Sensor parasitic capacitance
CRST	Counter reset
C _S	Capacitance of ADC measured
CTC	Capacitive touch controller
DCLK	Dot Clock
DDR	Double Data Rate
DE Mode	Data Enable Mode
DIF	Difference
DIR	Direction
DP	Duplicate
DRAM	Dynamic Random Access Memory; high-speed DRAM adds a separate clock signal to control signals.
DUT	Device Under Test
ECC	Error Correcting Code
EFT	Electrical fast transients
EMC	External Match Control, used to control the output status
EMC	Electro Magnetic Compatibility
ESD	Electrostatic discharge
ETC	Environment sensor capacitance tracking and calibration

FIFO	First In First Out
FMODE	FIFO Mode
FPU	Float Point Unit
FSM	Finite State Machine
Full-Functional Mode	A simulation model that describes the complete range of device behavior, including code execution. See also BFM .
GDMA	General Direct Memory Access
GF	Glitch Filter
GP	Group. There are two groups: group0 and group1. Hardware will automatically extract the corresponding MRs according TX_DATA[0]
GPIO	General Purpose Input/Output.
GRAM	Graphic Random-Access Memory
HBM	Human Body Model
HBP	Horizontal Back Porch
HDL	Hardware Description Language – examples include Verilog and VHDL.
HFP	Horizontal Front Porch
HSW	Horizontal Pulse Width
HSYNC	Horizontal Synchronization
HV Mode	Horizontal Vertical Mode
HWJ	Hardware judgement mode
I2S	Inter-IC Sound
INT	Interrupt
Interface	Set of ports and parameters that defines a connection point to a component.
IP	Intellectual property — A term that encompasses simulation models and synthesizable blocks of HDL code.
IR	Infrared
KV	Surge Voltage (KV)
LAT	Latch
LCDC	Liquid Crystal Display Controller
LCD-TFT	Liquid Crystal Display-Thin Film Transistor
LCM	Liquid Crystal Module
LD	Liquid Drop
LED	Light Emitting Diode
LP	Low power
LPC	Low Pin Count
master	Device or model that initiates and controls another device or peripheral.
MCNT	Match Counter, used to counter the match event
MCU	Micro CPU
MCU I/F	Micro Control Unit Interface
MCU Mode	Micro Control Unit Interface Mode

MPU	Memory Protection Unit
MR	Match Register, event(s) will be triggered following the configuration when the value of MR equals to the value of RXTX or MULTX
MUL	Multiple, can be used to realize Rx or Tx.
NVIC	Nested Vectored Interrupt Controller
OE	Output Enable
PDM	Pulse Density Modulation
peripheral	Generally refers to a small core that has a bus connection, specifically an APB interface.
PIR	Passive infrared
POS	Position, used for the control of shift operation.
PRTC	Current Value of Prescale Timer Counter
PRVAL	Maximum Value of Prescale, used for frequency division.
PSRAM	Pseudo Static Random Access Memory
QVGA	Quarter VGA
RC	Rotation Counter
RGB	Red-Green-Blue
RGB I/F	RGB Interface
RGB Mode	RGB Interface Mode
RGB565	Red (5 bits) - Green (6 bits) - Blue (5 bits)
RTL	Register Transfer Level. A higher level of abstraction that implies a certain gate-level structure.
RWDS	Read Write Data Strobe
SFT	Shift, indicate the generation of shift clock.
SGPIO	Serial GPIO
SI	Serial Interface
slave	Device or model that is controlled by and responds to a master.
SNR	Signal-to-noise ratio
SoC	System on a chip
SPORT	Serial PORT
SRAM	Static Random Access Memory
SRC	Source
SWD	Serial Wire Debug
SWJ	Software judgement mode
TC	Timer Counter
TE Mode	Tear Effect Mode
TMR	Timer
TVS	Transient voltage suppressors
UART	Universal Asynchronous Receiver/Transmitter
VBP	Vertical Back Porch

VFP	Vertical Front Porch
VGA	Video Graphics Array
VSW	VSYNC Pulse Width
VSNC	Vertical Synchronization

Revision History

Date	Version	Description
2020-01-13	v09	<ul style="list-style-type: none"> ● Update the MCU interface type of LCDC ● Update the section: Audio Pad
2019-09-20	v08	Update the support resolution of LCDC
2019-06-03	v07	<ul style="list-style-type: none"> ● Update the table of pad type ● Update the configurable bits of interrupt priority in NVIC
2019-04-10	v06	<ul style="list-style-type: none"> ● Optimize the technical expression ● Update the incorrect information
2019-02-21	v05	<ul style="list-style-type: none"> ● Add note about the state of GPIOs and time consumption in the process of system power on ● Normalize the technical expression ● Update some incorrect description ● Fill the missing content
2019-01-10	v04	<ul style="list-style-type: none"> ● Integrate the context of package types and pin description to Datasheet ● Integrate the context of boot process to Application Note ● Update the description of Product ● Update the feature list in LCDC ● Add Audio Pad in Pad Control and Pinmux ● Update some incorrect description and delete the compensation example in IR ● Update features/electrical characteristics/DMIC/AMIC connection figures in ACC ● Update the bit description of several registers in Q-Decoder and Key-Scan ● Update the MPU entry number of KM0 ● Add error rate table of baud rate in UART
2018-11-07	v03	<ul style="list-style-type: none"> ● Modify the incorrect information ● Update Wi-Fi calibration in eFuse ● Delete the specific descriptions about how to implement software handshaking ● Add Supported Resolution and update feature list in LCDC ● Delete section: Rx Path/Tx Path in UART
2018-09-21	v02	<ul style="list-style-type: none"> ● Unify and normalize the technical items and expression ● Modify the incorrect description ● Add necessary information for registers
2018-08-31	v01	Initial draft