

REALTEK

Ameba-Z II User Manual



REALTEK

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

©2019 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

Contents

Contents	3
List of Tables	20
List of Figures.....	31
History	35
1 Product Overview.....	36
1.1 General Description.....	36
1.2 System Architecture	37
1.3 Security Architecture.....	38
1.4 Power Architecture	39
2 Memory Organization	40
2.1 Overview	40
2.2 Internal ROM	40
2.3 Real-M300 Embedded SRAM	41
2.4 Retention Memory	41
2.5 SPI-NOR Flash Memory	41
2.6 pSRAM.....	42
2.7 eFuse	42
3 Power Mode and Control	44
3.1 Overview	44
3.1.1 Application Scenario.....	44
3.1.2 Features.....	45
3.1.3 Power Mode and Power Consumption.....	45
3.1.3.1 Overview of Power Modes	46
3.1.3.2 Clock-gated.....	47
3.1.3.3 Power-gated	47
3.2 Memory Control.....	48
3.3 Hardware Status of Power Modes	48
3.4 Shutdown Mode	50
3.5 Deep Sleep Mode	50
3.5.1 Wakeup Source.....	51
3.6 Standby Mode	52
3.6.1 Wakeup Source.....	52
3.7 Sleep Mode	53
3.7.1 Wakeup Source.....	54
3.8 Snooze Mode.....	55

3.8.1	Wakeup Source.....	56
4	Security Mode and Control.....	57
4.1	System Booting.....	57
4.2	Secure Boot	57
4.2.1	Firmware Image Encryption and Verification.....	58
4.2.2	Chain of Security	58
4.3	Ameba-Z II Secure features Overview	61
4.3.1	Functional Description	61
4.3.1.1	Hardware Crypto Accelerator.....	61
4.3.1.2	Flash Memory Protection.....	61
4.3.1.3	Debugger (JTAG/SWD) Protection.....	62
4.4	Device Lifecycle State Management	63
5	Pad Control and Pinmux.....	66
5.1	Pad Control.....	66
5.1.1	Pad Types.....	66
5.1.2	Pad Pull Control	67
5.1.3	Pad Schmitt Trigger	67
5.1.4	Pad Driving Strength.....	67
5.1.5	Pad Shut Down	68
5.2	Pin Multiplexing Function	68
6	Real-M300 (KM4) CPU.....	70
6.1	Tightly Coupled Memory (TCM)	71
6.2	CPU I/D Cache Engine.....	73
6.2.1	Introduction.....	73
6.2.2	Function Description	73
6.3	CPU System Tick Timer	74
6.3.1	Overview	74
6.3.1.1	Introduction.....	74
6.3.1.2	Features.....	75
6.3.2	Functional Description	75
6.3.2.1	Overview	75
6.4	Nested Vectored Interrupt Controller (NVIC).....	75
6.4.1	Overview	75
6.4.1.1	Features.....	76
6.4.1.2	NVIC Table	76
6.4.2	NVIC Control Registers	77
6.5	Memory Protection Unit (MPU).....	77
6.5.1	Overview	77

6.5.2	MPU Control Registers	77
7	General Timer (GTIMER)	79
7.1	HS Timer	79
7.1.1	Overview	79
7.1.1.1	Application Scenario.....	79
7.1.1.2	Features.....	79
7.1.1.3	Architecture.....	80
7.1.2	Functional Description	81
7.1.2.1	Mode Selection	81
7.1.2.2	Counting Mode.....	81
7.1.2.3	Match Event	82
7.1.2.4	Interrupt Event	82
7.1.3	Control Registers	83
7.1.3.1	TG0_ISTS.....	83
7.1.3.2	TG0_RAW_ISTS.....	83
7.1.3.3	TG0_Tmir_CTRL.....	83
7.1.3.4	TG0_Tmir_TC.....	84
7.1.3.5	TM0_LC.....	84
7.1.3.6	TM0_TC.....	84
7.1.3.7	TM0_PC	84
7.1.3.8	TM0_PR	85
7.1.3.9	TM0_CTRL	85
7.1.3.10	TM0_ISR	86
7.1.3.11	TM0_MECTRL	86
7.1.3.12	TM0_ME0	86
7.1.3.13	TM0_ME1	87
7.1.3.14	TM0_ME2	87
7.1.3.15	TM0_ME3	87
7.1.3.16	TMx_Related	87
7.2	LP Timer.....	88
7.2.1	Overview	88
7.2.1.1	Application Scenario.....	88
7.2.1.2	Features.....	88
7.2.1.3	Architecture.....	88
7.2.2	Functional Description	89
7.2.2.1	Mode Selection	90
7.2.2.2	Counting Mode.....	90
7.2.2.3	Match Event	90

7.2.2.4	Interrupt Event	91
7.2.3	Control Registers	91
7.2.3.1	TG0_ISTS	91
7.2.3.2	TG0_RAW_ISTS	92
7.2.3.3	TG0_T米尔_CTRL	92
7.2.3.4	TG0_T米尔_TC	92
7.2.3.5	TM0_LC	93
7.2.3.6	TM0_TC	93
7.2.3.7	TM0_PC	93
7.2.3.8	TM0_PR	94
7.2.3.9	TM0_CTRL	94
7.2.3.10	TM0_ISR	94
7.2.3.11	TM0_MECTRL	95
7.2.3.12	TM0_MEO	95
7.2.3.13	TM0_ME1	96
7.2.3.14	TM0_ME2	96
7.2.3.15	TM0_ME3	96
8	Watchdog Timer (WDT)	97
8.1	Overview	97
8.1.1	Application Scenario	97
8.1.2	Features	97
8.1.3	Architecture	98
8.2	Functional Description	98
8.2.1	Operation Mode	98
8.3	Control Registers	99
8.3.1	WATCH_DOG_TIMER	99
9	Pulse Width Modulation (PWM)	100
9.1	Overview	100
9.1.1	Application Scenario	100
9.1.2	Features	100
9.1.3	Architecture	101
9.2	Functional Description	102
9.2.1	PWM Mode	102
9.2.2	PWM Auto Adjustment	103
9.2.3	PWM Interrupt Event	103
9.3	Control Registers	104
9.3.1	PWM_EN_STS	104
9.3.2	PWM_EN	105

9.3.3	PWM_DIS	105
9.3.4	PWM_INT_STATUS	105
9.3.5	PWM_INDREAD_IDX	107
9.3.6	PWM_INDREAD_DUTY	107
9.3.7	PWMx_CTRL	108
9.3.8	PERI_PWMx_TIM_CTRL.....	108
9.3.9	PWMx_DUTY_AUTO_ADJ_CTRL.....	109
9.3.10	PWMx_DUTY_AUTO_ADJ_LIMIT.....	110
9.3.11	PWM_DUTY_AUTO_ADJ_CYCLE.....	110
10	Universal Asynchronous Receiver/ Transmitter (UART)	111
10.1	Overview	111
10.1.1	Application Scenario.....	111
10.1.2	Features.....	111
10.1.3	Architecture.....	111
10.1.3.1	Functional Block Diagram with Clock Domain	112
10.2	Functional Description	113
10.2.1	Overview.....	113
10.2.2	UART Terminology	113
10.2.2.1	UART Data Format	113
10.2.3	Baud Rate Calculation	114
10.2.4	Auto Flow Control	115
10.2.5	RX Filter	116
10.3	Control Registers	118
10.3.1	DLLR	118
10.3.2	DLMR	118
10.3.3	IER	119
10.3.4	IIR	119
10.3.5	FCR	121
10.3.6	LCR	122
10.3.7	MCR	124
10.3.8	LSR	126
10.3.9	MSR	128
10.3.10	SCR	128
10.3.11	STSRA	129
10.3.12	RBR	130
10.3.13	THR	130
10.3.14	MISCCR	131
10.3.15	IRDA_TXPLSR	131

10.3.16	DBG_UART.....	132
10.3.17	RFCR.....	133
10.3.18	RFMPR	133
10.3.19	RFMVR.....	134
10.3.20	RFTOR	135
10.3.21	RFLVR.....	135
10.3.22	TFLVR	136
10.3.23	VDR_INTSR	136
10.3.24	VDR_INTER	137
10.3.25	RXIDLE_TOCR.....	137
11	Serial Peripheral Interface (SPI).....	139
11.1	Overview	139
11.1.1	Application Scenario.....	139
11.1.2	Features.....	139
11.1.3	Architecture.....	140
11.1.3.1	Functional Block Diagram.....	141
11.2	Functional Description	142
11.2.1	Overview	142
11.2.2	SPI Transfer Protocol	143
11.2.2.1	Motorola Serial Peripheral Interface (SPI).....	143
11.2.2.2	Texas Instruments Synchronous Serial Protocol (SSP)	145
11.2.3	SPI Interrupts.....	145
11.2.3.1	Transmit FIFO Empty Interrupt (ssi_txe_intr).....	145
11.2.3.2	Transmit FIFO Overflow Interrupt (ssi_txo_intr)	145
11.2.3.3	Receive FIFO Full Interrupt (ssi_rxf_intr)	146
11.2.3.4	Receive FIFO Overflow Interrupt (ssi_rxo_intr).....	146
11.2.3.5	Receive FIFO Underflow Interrupt (ssi_rxu_intr)	146
11.2.3.6	Multi-Master Contention Interrupt (ssi_mst_intr).....	146
11.2.3.7	Transmit FIFO Underflow Interrupt (ssi_txu_intr).....	146
11.2.3.8	SS_N Rising Edge Detect Interrupt (ssi_icr_intr)	146
11.2.3.9	Combined Interrupt Request (ssi_intr)	147
11.3	Control Registers	147
11.3.1	SPI_CTRLR0.....	147
11.3.2	SPI_CTRLR1.....	148
11.3.3	SPI_SSIENR.....	149
11.3.4	SPI_MWCR.....	149
11.3.5	SPI_SER.....	150
11.3.6	SPI_BAUDR	151

11.3.7	SPI_TXFTLR	151
11.3.8	SPI_RXFTLR	152
11.3.9	SPI_TXFLR	152
11.3.10	SPI_RXFLR	153
11.3.11	SPI_SR	153
11.3.12	SPI_IMR	154
11.3.13	SPI_ISR	155
11.3.14	SPI_RISR	156
11.3.15	SPI_TXOICR	156
11.3.16	SPI_RXOICR	157
11.3.17	SPI_RXUICR	157
11.3.18	SPI_MSTICR	157
11.3.19	SPI_ICR	158
11.3.20	SPI_DMACR	158
11.3.21	SPI_DMATDLR	159
11.3.22	SPI_DMARDLR	159
11.3.23	SPI_TXUICR	159
11.3.24	SPI_SSRICR	160
11.3.25	SPI_DR	160
12	Inter-Integrated Circuit Interface (I2C)	161
12.1	Overview	161
12.1.1	Application Scenario	161
12.1.2	Features	161
12.1.3	Architecture	162
12.1.3.1	Functional Block Diagram with Clock Domain	162
12.2	Functional Description	163
12.2.1	Overview	163
12.2.2	I2C Terminology	163
12.2.2.1	I2C Bus Terms	163
12.2.2.2	Bus Transfer Terms	164
12.2.3	I2C Behavior	165
12.2.3.1	START and STOP Generation	166
12.2.3.2	Combined Formats	166
12.2.4	I2C Protocols	166
12.2.4.1	START and STOP Conditions	167
12.2.4.2	Addressing Slave Protocol	167
12.2.4.3	Transmitting and Receiving Protocol	168
12.2.5	Tx FIFO Operation and Control	169

12.2.5.1	DMA Operation	171
12.3	Control Registers	172
12.3.1	IC_CON	172
12.3.2	IC_TAR	173
12.3.3	IC_SAR	173
12.3.4	IC_HS_MADDR	174
12.3.5	IC_DATA_CMD	174
12.3.6	IC_SS_SCL_HCNT	175
12.3.7	IC_SS_SCL_LCNT	175
12.3.8	IC_FS_SCL_HCNT	175
12.3.9	IC_FS_SCL_LCNT	176
12.3.10	IC_HS_SCL_HCNT	176
12.3.11	IC_HS_SCL_LCNT	176
12.3.12	IC_INTR_STAT	177
12.3.13	IC_INTR_MASK	177
12.3.14	IC_RAW_INTR_STAT	178
12.3.15	IC_RX_TL	179
12.3.16	IC_TX_TL	179
12.3.17	IC_CLR_INTR	180
12.3.18	IC_CLR_RX_UNDER	180
12.3.19	IC_CLR_RX_OVER	180
12.3.20	IC_CLR_TX_OVER	181
12.3.21	IC_CLR_RD_REQ	181
12.3.22	IC_CLR_TX_ABRT	181
12.3.23	IC_CLR_RX_DONE	182
12.3.24	IC_CLR_ACTIVITY	182
12.3.25	IC_CLR_STOP_DET	183
12.3.26	IC_CLR_START_DET	183
12.3.27	IC_CLR_GEN_CALL	183
12.3.28	IC_ENABLE	184
12.3.29	IC_STATUS	184
12.3.30	IC_TXFLR	185
12.3.31	IC_RXFLR	185
12.3.32	IC_SDA_HOLD	185
12.3.33	IC_TX_ABRT_SOURCE	186
12.3.34	IC_SLV_DATA_NACK_ONLY	188
12.3.35	IC_DMA_CR	188
12.3.36	IC_DMA_TDRL	188

12.3.37	IC_DMA_RDLR.....	189
12.3.38	IC_SDA_SETUP.....	189
12.3.39	IC_ACK_GENERAL_CALL	190
12.3.40	IC_ENABLE_STATUS	190
12.3.41	IC_DMA_CMD	190
12.3.42	IC_DMA_LEN	191
12.3.43	IC_DMA_MOD	191
12.3.44	IC_SLEEP	192
12.3.45	IC_DEBUG_SEL.....	192
12.3.46	IC_STRCH_DLY	192
12.3.47	IC_ANA_FLTR_SDA_RL.....	193
12.3.48	IC_ANA_FLTR_SDA_RM	193
12.3.49	IC_ANA_FLTR_SCL_RL.....	193
12.3.50	IC_ANA_FLTR_SCL_RM	194
12.3.51	IC_ANA_FLTR_SDA_CL	194
12.3.52	IC_ANA_FLTR_SDA_CM	194
12.3.53	IC_ANA_FLTR_SCL_CL	195
12.3.54	IC_ANA_FLTR_SCL_CM	195
12.3.55	IC_CLR_DMA_I2C_DONE.....	195
12.3.56	IC_FILTER	196
12.3.57	IC_SAR1	196
12.3.58	IC_COMP_VER	196
13	Realtek DMA Controller (DMAC)	198
13.1	Overview	198
13.1.1	Application Scenario.....	198
13.1.2	Features.....	198
13.1.3	Architecture.....	199
13.1.3.1	Functional Block Diagram.....	200
13.2	Functional Description.....	201
13.2.1	Overview	201
13.2.2	DMA Hardware Handshake	202
13.2.3	Peripheral Burst Transaction Requests.....	203
13.2.4	RTK-DMAC Interrupts	205
13.2.4.1	Block Transfer Complete Interrupt (IntBlock)	205
13.2.4.2	Destination Transaction Complete Interrupt (IntDstTran)	205
13.2.4.3	Source Transaction Complete Interrupt (IntSrcTran)	205
13.2.4.4	Error Interrupt (IntErr).....	205
13.2.4.5	DMA Transfer Complete Interrupt (IntTfr)	205

13.3	Control Registers	206
13.3.1	DMA_SARx.....	206
13.3.2	DMA_DARx.....	206
13.3.3	DMA_CTLx	207
13.3.4	DMA_CFGx	208
13.3.5	DMA_RawTfr	210
13.3.6	DMA_RawBlock	210
13.3.7	DMA_RawSrcTran.....	211
13.3.8	DMA_RawDstTran	211
13.3.9	DMA_RawErr	211
13.3.10	DMA_StatusTfr	212
13.3.11	DMA_StatusBlock.....	212
13.3.12	DMA_StatusSrcTran.....	212
13.3.13	DMA_StatusDstTran	213
13.3.14	DMA_StatusErr	213
13.3.15	DMA_MaskTfr	213
13.3.16	DMA_MaskBlock	214
13.3.17	DMA_MaskSrcTran	215
13.3.18	DMA_MaskDstTran.....	215
13.3.19	DMA_MaskErr	216
13.3.20	DMA_ClearTfr.....	217
13.3.21	DMA_ClearBlock.....	217
13.3.22	DMA_ClearSrcTran	217
13.3.23	DMA_ClearDstTran	218
13.3.24	DMA_ClearErr.....	218
13.3.25	DMA_StatusInt	219
13.3.26	DMA_DmaCfgReg	219
13.3.27	DMA_ChEnReg	220
13.3.28	DMA_SoftResetReg	220
14	SDIO Device/SPI Slave Interface	222
14.1	Overview	222
14.1.1	Application Scenario.....	222
14.1.2	Features.....	223
14.1.3	Architecture.....	224
14.1.3.1	SDIO Device Mode Architecture.....	225
14.1.3.2	SPI Slave Mode Architecture	226
14.2	Functional Description	227
14.2.1	SPDIO DMA Transfer.....	229

14.2.1.1	TX Flow	229
14.2.1.2	RX Flow	232
14.2.2	SDIO Device Interface Control	235
14.2.3	SPI Slave Interface	238
14.2.3.1	SPI Interface Protocol	240
14.3	SDIO Control Registers	245
14.3.1	SDIO_TX_CTRL.....	245
14.3.2	SDIO_STATIS_RECOVERY_TIMEOUT	245
14.3.3	SDIO_HIMR	246
14.3.4	SDIO_HISR	246
14.3.5	RX0_REQ_LEN.....	247
14.3.6	FREE_TXBD_NUM.....	247
14.3.7	TX_SEQNUM.....	248
14.3.8	HCPWM	248
14.3.9	HCPWM2	248
14.3.10	REG_SDIO_H2C_MSG	249
14.3.11	REG_SDIO_C2H_MSG	249
14.3.12	REG_SDIO_H2C_MSG_EXT	249
14.3.13	REG_SDIO_C2H_MSG_EXT	250
14.3.14	HRPWM	250
14.3.15	HRPWM2	250
14.3.16	HPS_CLKR	251
14.3.17	CPU_INDICATION	251
14.3.18	CMD_IN_TO_RSP_OUT_TIME	251
14.3.19	ERR_FLAG	252
14.3.20	SDIO_CMD_ERRCNT	252
14.3.21	SDIO_DATA_ERRCNT	252
14.3.22	SDIO_CRC_ERR_INDEX	253
14.3.23	SDIO_AVAI_BD_NUM_TH_L	253
14.3.24	SDIO_AVAI_BD_NUM_TH_H	253
14.3.25	SDIO_RX_AGG_CFG	254
14.3.26	SDIO_CMD_TIMING_CTRL	254
14.3.27	SD_DATA01_TIMING_CTRL	254
14.3.28	SD_DATA23_TIMING_CTRL	255
14.4	SPDIO Control Registers	255
14.4.1	SPDIO_TXBD_ADDR	255
14.4.2	SPDIO_TXBD_NUM	255
14.4.3	SPDIO_TXBD_WPTR	256

14.4.4	SPDIO_TXBD_RPTR.....	256
14.4.5	SPDIO_RXBD_ADDR	256
14.4.6	SPDIO_RXBD_NUM	257
14.4.7	SPDIO_RXBD_C2H_WPTR.....	257
14.4.8	SPDIO_RXBD_C2H_RPTR.....	257
14.4.9	HCI_RX_REQ	258
14.4.10	CPU_RST_SDIO_DMA.....	258
14.4.11	SPDIO_RX_REQ_ADDR	258
14.4.12	SPDIO_CPU_INT_MASK.....	259
14.4.13	SPDIO_CPU_INT_REG.....	260
14.4.14	CCPWM	261
14.4.15	SYS_CPU_INDICATION	261
14.4.16	CCPWM2	262
14.4.17	REG_CPU_H2C_MSG	262
14.4.18	REG_CPU_C2H_MSG	262
14.4.19	CRPWM	263
14.4.20	CRPWM2	263
14.4.21	AHB_DMA_CTRL.....	264
14.4.22	FREE_RXBD_COUNT	264
14.4.23	REG_CPU_H2C_MSG_EXT	265
14.4.24	REG_CPU_C2H_MSG_EXT	265
14.5	GSPI Control Registers	266
14.5.1	SPI_INT	266
14.5.2	SPI_HIMRxx	266
14.5.3	SPI_HISR	266
14.5.4	RX0_REQ_LEN.....	267
14.5.5	FREE_TX_BD_NUM.....	268
14.5.6	TX_SEQNUM.....	268
14.5.7	HCPWM	268
14.5.8	HCPWM2	269
14.5.9	SPI_AVAI_PGTH_L.....	269
14.5.10	SPI_AVAI_PGTH_H.....	269
14.5.11	SPI_RX_AGG	269
14.5.12	REG_SPI_H2C_MSG	270
14.5.13	REG_SPI_C2H_MSG	270
14.5.14	HRPWM	271
14.5.15	HPS_CLKR	271
14.5.16	CPU_INDICATION.....	271

14.5.17	32K_TRANSPARENT	272
14.5.18	32K_IDLE.....	272
14.5.19	DELY_LINE_SEL	272
14.5.20	SPI_CFG	273
15	SPI NOR Flash Controller (SPIC)	274
15.1	Overview	274
15.1.1	Application Scenario.....	274
15.1.2	Features.....	274
15.1.3	Architecture.....	275
15.1.3.1	Functional Block Diagram.....	276
15.2	Functional Description	277
15.2.1	Overview	277
15.2.2	SPIC Pin Connection	277
15.2.3	SPIC Transfer Protocol	278
15.2.4	SPIC Transfer Mode	279
15.2.4.1	Transmit Mode	280
15.2.4.2	Receive Mode.....	281
15.2.5	SPIC Operation Mode	281
15.2.5.1	User Mode.....	281
15.2.5.2	Auto Mode	282
15.2.6	Flash Calibration Mechanism	283
15.3	Control Registers	284
15.3.1	SPIC_CTRLR0.....	284
15.3.2	SPIC_CTRLR1.....	285
15.3.3	SPIC_SSIENR	285
15.3.4	SPIC_SER.....	285
15.3.5	SPIC_BAUDR	286
15.3.6	SPIC_TXFLR	286
15.3.7	SPIC_RXFLR.....	286
15.3.8	SPIC_SR.....	287
15.3.9	SPIC_IDR	287
15.3.10	SPIC_VERSION	288
15.3.11	SPIC_DR.....	288
15.3.12	SPIC_READ_FAST_SINGLE.....	288
15.3.13	SPIC_READ_DUAL_DATA.....	289
15.3.14	SPIC_READ_DUAL_ADDR_DATA	289
15.3.15	SPIC_READ_QUAD_DATA.....	290
15.3.16	SPIC_READ_QUAD_ADDR_DATA	290

15.3.17	SPIC_WRITE_SINGLE.....	290
15.3.18	SPIC_WRITE_ENABLE	291
15.3.19	SPIC_READ_STATUS	291
15.3.20	SPIC_CTRLR2.....	292
15.3.21	SPIC_FBAUDR	292
15.3.22	SPIC_ADDR_LENGTH	292
15.3.23	SPIC_AUTO_LENGTH	293
15.3.24	SPIC_VALID_CMD	294
15.3.25	SPIC_FLUSH_FIFO	295
16	General Purpose Input/Output (GPIO).....	296
16.1	Overview	296
16.1.1	Application Scenario.....	296
16.1.2	Features.....	296
16.1.3	Architecture.....	297
16.2	Functional Description.....	297
16.2.1	GPIO Input and Output Control.....	297
16.2.2	Interrupts.....	298
16.2.3	Debounce Operation	299
16.3	Control Registers	300
16.3.1	GPIO_IT_STS	300
16.3.2	GPIO_EI_EN	300
16.3.3	GPIO_LI_EN	300
16.3.4	GPIO_IP_STS.....	301
16.3.5	GPIO_IR_EN.....	303
16.3.6	GPIO_IF_EN	303
16.3.7	GPIO_ID_EN.....	303
16.3.8	GPIO_IM_STS.....	304
16.3.9	GPIO_IM_EN.....	304
16.3.10	GPIO_IM_DIS.....	304
16.3.11	GPIO_RAW_INT_STS.....	305
16.3.12	GPIO_INT_STS	305
16.3.13	GPIO_INT_CLR	305
16.3.14	GPIO_INT_EN_STS	306
16.3.15	GPIO_INT_EN	306
16.3.16	GPIO_INT_DIS	306
16.3.17	GPIO_INTO_SEL.....	307
16.3.18	GPIO_INT1_SEL.....	307
16.3.19	GPIO_INT2_SEL.....	308

16.3.20	GPIO_INT3_SEL.....	308
16.3.21	GPIO_INT4_SEL.....	308
16.3.22	GPIO_INT5_SEL.....	309
16.3.23	GPIO_INT6_SEL.....	309
16.3.24	GPIO_INT7_SEL.....	310
16.3.25	GPIO_INT8_SEL.....	310
16.3.26	GPIO_INT9_SEL.....	311
16.3.27	GPIO_INTA_SEL.....	311
16.3.28	GPIO_INTB_SEL	311
16.3.29	GPIO_INTC_SEL.....	312
16.3.30	GPIO_INTD_SEL	312
16.3.31	GPIO_INTE_SEL.....	313
16.3.32	GPIO_INTF_SEL.....	313
16.3.33	GPIO_DEB_STS.....	314
16.3.34	GPIO_DEB_EN	314
16.3.35	GPIO_DEB_DIS.....	314
16.3.36	DEB_DP_STS	315
16.3.37	GPIO_DEB0_SEL	315
16.3.38	GPIO_DEB1_SEL	315
16.3.39	GPIO_DEB2_SEL	316
16.3.40	GPIO_DEB3_SEL	316
16.3.41	GPIO_DEB4_SEL	316
16.3.42	GPIO_DEB5_SEL	317
16.3.43	GPIO_DEB6_SEL	317
16.3.44	GPIO_DEB7_SEL	318
16.3.45	GPIO_DEB8_SEL	318
16.3.46	GPIO_DEB9_SEL	318
16.3.47	GPIO_DEBA_SEL	319
16.3.48	GPIO_DEBB_SEL	319
16.3.49	GPIO_DEBC_SEL	320
16.3.50	GPIO_DEBD_SEL	320
16.3.51	GPIO_DEBE_SEL.....	320
16.3.52	GPIO_DEBF_SEL.....	321
16.3.53	GPIOA_DMD_STS	321
16.3.54	GPIOA_IDM_EN	322
16.3.55	GPIOA_ODM_EN	322
16.3.56	GPIOA_OD_STS.....	322
16.3.57	GPIOA_ODL_EN	323

16.3.58	GPIOA_ODH_EN	323
16.3.59	GPIOA_ODT_EN.....	323
16.3.60	GPIOA_DP_STS	324
17	Security Code Engine (SCE)	325
17.1	Overview	325
17.1.1	Application Scenario.....	325
17.1.2	Features.....	325
17.1.3	Architecture.....	325
17.2	Functional Description	326
17.2.1	Decrypt Control Flow	326
17.2.2	Remap Function	328
18	Crypto Engine	329
18.1	Overview	329
18.1.1	Application Scenario.....	329
18.1.2	Features.....	329
18.1.3	Architecture.....	330
18.1.3.1	Functional Block Diagram with Clock Domain	331
18.2	Functional Description	332
18.2.1	Overview	332
18.2.2	Descriptor Setting Mechanism	332
18.2.2.1	Descriptor FIFO Architecture.....	332
18.2.3	Descriptor Data Structures	336
18.2.3.1	Source Crypto Descriptor Type1.....	336
18.2.3.2	Source Crypto Descriptor Command Setting	337
18.2.3.3	Source Crypto Descriptor Type2.....	342
18.2.3.4	Destination Crypto Descriptor Type1	343
18.2.3.5	Destination Crypto Descriptor Type2	343
18.3	Control Registers	344
18.3.1	SDSR	344
18.3.2	SDFWR	345
18.3.3	SDSWR	345
18.3.4	IPCSR_CONF_INT	346
18.3.5	IPCSR_INT_MASK.....	347
18.3.6	IPCSR_DBG_DMA	348
18.3.7	IPCSR_ERR_STATUS.....	349
18.3.8	IPCSR_SUM_OF_AADL.....	351
18.3.9	IPCSR_SUM_OF_EDL	351
18.3.10	IPCSR_SUM_OF_APDL.....	352

18.3.11	IPCSR_SUM_OF_EPDL	352
18.3.12	IPCSR_SWAP_BURST.....	352
18.3.13	DDSR.....	353
18.3.14	DDFWR	354
18.3.15	DDSWR	354
18.3.16	IPCSR_CONF_PKTABR	355
18.3.17	IPCSR_DBG_SDDR.....	355
18.3.18	ICSR_DBG_DDDR	356
19	Wi-Fi	357
19.1	Overview	357
19.1.1	Application Scenario.....	357
19.1.2	Features.....	357
19.1.3	Architecture.....	358
19.2	Functional Description	359
19.2.1	Overview	359
19.2.2	Wi-Fi Unit Transmit/Receive Flow	359
19.2.2.1	Overview	359
19.2.2.2	Wi-Fi MAC TRx Flow	360
19.2.3	Wi-Fi Power Save Mode	360
19.2.3.1	Overview	360
19.2.3.2	The State Machine of The Power Save Mode	361
19.2.3.3	The Behavior of The Power Save Mode	361
19.2.4	Wi-Fi Wake On Wireless LAN.....	362
19.2.4.1	Overview	362
19.2.4.2	The Behavior of The Wake On Wireless LAN.....	363
20	Bluetooth.....	364
20.1	Overview	364
20.1.1	Features.....	364
20.1.2	BLE Profile Architecture.....	364
20.1.2.1	GAP	365
20.1.2.2	GATT Based Profile	365
Appendix A:	Abbreviations.....	367
Appendix B:	ARM ACK Certificate	370

List of Tables

Table 2-1 Address Mapping	40
Table 2-2 Description of Flash Memory Layout.....	42
Table 3-1 Power Consumption	48
Table 3-2 Hardware Status of Power Mode	49
Table 3-3 Deep Sleep Mode Wakeup Source	51
Table 3-4 Standby Mode Wakeup Source	52
Table 3-5 Sleep Mode Wakeup Source.....	54
Table 5-1 Pad Types	66
Table 5-2 Pad Pull Control	67
Table 5-3 Characteristics of Schmitt Trigger	67
Table 5-4 Pad Driving Control of Normal and SDIO Pad.....	67
Table 5-5 Pad Driving Control of 5V Pad	68
Table 5-6 Pad Shut Down Control.....	68
Table 5-7 GPIOA0_PINMUX_SEL	68
Table 5-8 Pin Multiplexing Table	69
Table 6-1 The Performance Table of Real-M300 vs CM33	70
Table 6-2 TCM Related Registers	72
Table 6-3 Real-M300 Data Read/Write Priority over TCMs and Caches	72
Table 6-4 TCM Filling Control in ACTLR	73
Table 6-5 Real-M Cache Behaviors	74
Table 6-6 NVIC Table.....	76
Table 6-7 NVIC Related Control Registers.....	77
Table 6-8 MPU Related Control Registers.....	78
Table 6-9 Recommended MPU Attrfield setting in MAIR0 and MAIR1.....	78
REG 7-1 (Offset 0000h) TG0ISTS	83
REG 7-2 (Offset 0004h) TG0_RAWISTS	83
REG 7-3 (Offset 0008h) TG0_TMIR_CTRL	83
REG 7-4 (Offset 000Ch) TG0_TMIR_TC	84
REG 7-5 (Offset 0040h) TM0LC	84
REG 7-6 (Offset 0044h) TM0TC	84
REG 7-7 (Offset 0048h) TM0PC	85
REG 7-8 (Offset 004Ch) TM0PR	85
REG 7-9 (Offset 0050h) TM0CTRL	85
REG 7-10 (Offset 0054h) TM0ISR	86
REG 7-11 (Offset 005Ch) TM0MECTRL	86

REG 7-12 (Offset 0060h) TM0_ME0	87
REG 7-13 (Offset 0064h) TM0_ME1	87
REG 7-14 (Offset 0068h) TM0_ME2	87
REG 7-15 (Offset 006Ch) TM0_ME3	87
REG 7-16 (Offset (x+1)*0x40h) TMX_Related.....	88
REG 7-17 (Offset 0000h) TG0_ISTS.....	92
REG 7-18 (Offset 0004h) TG0_RAW_ISTS	92
REG 7-19 (Offset 0008h) TG0_Tmir_CTRL	92
REG 7-20 (Offset 000Ch) TG0_Tmir_TC	93
REG 7-21 (Offset 0040h) TM0_LC	93
REG 7-22 (Offset 0044h) TM0_TC	93
REG 7-23 (Offset 0048h) TM0_PC	93
REG 7-24 (Offset 004Ch) TM0_PR	94
REG 7-25 (Offset 0050h) TM0_CTRL	94
REG 7-26 (Offset 0054h) TM0_ISR	95
REG 7-27 (Offset 005Ch) TM0_MECTRL	95
REG 7-28 (Offset 0060h) TM0_ME0	95
REG 7-29 (Offset 0064h) TM0_ME1	96
REG 7-30 (Offset 0068h) TM0_ME2	96
REG 7-31 (Offset 006Ch) TM0_ME3	96
REG 8-1 (Offset 0000h) WATCH_DOG_TIMER	99
REG 9-1 (Offset 0000h) PWM_EN_STS	104
REG 9-2 (Offset 0004h) PWM_EN	105
REG 9-3 (Offset 0008h) PWM_DIS	105
REG 9-4 (Offset 000Ch) PWM_INT_STATUS	105
REG 9-5 (Offset 0010h) PWM_INDREAD_IDX	107
REG 9-6 (Offset 0014h) PWM_INDREAD_DUTY	107
REG 9-7 (Offset 000h + (0x20h * (x+1))) PWMx_CTRL	108
REG 9-8 (Offset 004h + (0x20h * (x+1))) PWMx_TIM_CTRL	109
REG 9-9 (Offset 008h + (0x20h * (x+1))) PWMx_DUTY_AUTO_ADJ_CTRL	109
REG 9-10(Offset 00Ch + (0x20h * (x+1))) PWMx_DUTY_AUTO_ADJ_LIMIT	110
REG 9-11 (Offset 010h + (0x20h * (x+1))) PWMx_DUTY_AUTO_ADJ_CYCLE	110
Table 10-1 Baud Rate Error Calculation	115
Table 10-2 Error Rate of Baud Rate	115
REG 10-3 (Offset 0000h) DLLR	118
REG 10-4 (Offset 0004h) DLMR	118
REG 10-5 (Offset 0004h) IER	119
REG 10-6 (Offset 0008h) IIR	120

Table 10-7 Interrupt Control Functions	121
REG 10-8 (Offset 0008h) FCR	122
REG 10-9 (Offset 000Ch) LCR	123
REG 10-10 (Offset 0010h) MCR	124
REG 10-11 (Offset 0014h) LSR	126
REG 10-12 (Offset 0018h) MSR	128
REG 10-13 (Offset 001Ch) SCR	129
REG 10-14 (Offset 0028h) STSR	129
REG 10-15 (Offset 0024h) RBR	130
REG 10-16 (Offset 0024h) THR	130
REG 10-17 (Offset 0028h) MISCCR	131
Table 10-18 Signaling Rate and Pulse Duration Specification	132
REG 10-19 (Offset 002Ch) IRDA_TXPLSR	132
REG 10-20 (Offset 003Ch) DBG_UART	132
REG 10-21 (Offset 0040h) RFCR	133
REG 10-22 (Offset 0044h) RFMPR	134
REG 10-23 (Offset 0048h) RFMVR	134
Table 10-24 RX Filter Matching Condition	134
REG 10-25 (Offset 004Ch) RFTOR	135
REG 10-26 (Offset 0050h) RFLVR	135
REG 10-27 (Offset 0054h) TFLVR	136
REG 10-28 (Offset 0058h) VDR_INSTR	136
REG 10-29 (Offset 005Ch) VDR_INTER	137
REG 10-30 (Offset 0060h) RXIDLE_TOCR	137
Table 11-1 SPI Speed	139
Table 11-2 SPI Clock Polarity	140
REG 11-3 (Offset 0000h) SPI_CTRLR0	147
REG 11-4 (Offset 0004h) SPI_CTRLR1	149
REG 11-5 (Offset 0008h) SPI_SSIENR	149
REG 11-6 (Offset 000Ch) SPI_MWCR	150
REG 11-7 (Offset 0010h) SPI_SER	150
REG 11-8 (Offset 0014h) SPI_BAUDR	151
REG 11-9 (Offset 0018h) SPI_TXFTLR	151
Table 11-10 Transmit FIFO Threshold Value Decode Field	151
REG 11-11 (Offset 001Ch) SPI_RXFTLR	152
Table 11-12 Receive FIFO Threshold Value Decode Field	152
REG 11-13 (Offset 0020h) SPI_TXFLR	152
REG 11-14 (Offset 0024h) SPI_RXFLR	153

REG 11-15 (Offset 0028h) SPI_SR.....	153
REG 11-16 (Offset 002Ch) SPI_IMR	154
REG 11-17 (Offset 0030h) SPI_ISR.....	155
REG 11-18 (Offset 0034h) SPI_RISR.....	156
REG 11-19 (Offset 0038h) SPI_TXOICR	157
REG 11-20 (Offset 003Ch) SPI_RXOICR.....	157
REG 11-21 (Offset 0040h) SPI_RXUICR.....	157
REG 11-22 (Offset 0044h) SPI_MSTICR	158
REG 11-23 (Offset 0048h) SPI_ICR.....	158
REG 11-24 (Offset 004Ch) SPI_DMACR	158
REG 11-25 (Offset 0050h) SPI_DMATDLR.....	159
REG 11-26 (Offset 0054h) SPI_DMARDLR	159
REG 11-27 (Offset 0058h) SPI_TXUICR	160
REG 11-28 (Offset 005Ch) SPI_SSRICR.....	160
REG 11-29 (Offset 0060h) SPI_DR	160
REG 12-1 (Offset 0000h) IC_CON	172
REG 12-2 (Offset 0004h) IC_TAR.....	173
REG 12-3 (Offset 0008h) IC_SAR	173
REG 12-4 (Offset 000Ch) IC_HS_MADDR	174
REG 12-5 (Offset 0010h) IC_DATA_CMD	174
REG 12-6 (Offset 0014h) IC_SS_SCL_HCNT	175
REG 12-7 (Offset 0018h) IC_SS_SCL_LCNT	175
REG 12-8 (Offset 001Ch) IC_FS_SCL_HCNT	175
REG 12-9 (Offset 0020h) IC_FS_SCL_HCNT	176
REG 12-10 (Offset 0024h) IC_HS_SCL_HCNT	176
REG 12-11 (Offset 0028h) IC_HS_SCL_LCNT	176
REG 12-12 (Offset 002Ch) IC_INTR_STAT	177
REG 12-13 (Offset 0030h) IC_INTR_MASK	177
REG 12-14 (Offset 0034h) IC_RAW_INTR_STAT	178
REG 12-15 (Offset 0038h) IC_RX_TL.....	179
REG 12-16 (Offset 003Ch) IC_TX_TL.....	180
REG 12-17 (Offset 0040h) IC_CLR_INTR	180
REG 12-18 (Offset 0044h) IC_CLR_RX_UNDER.....	180
REG 12-19 (Offset 0048h) IC_CLR_RX_OVER	181
REG 12-20 (Offset 004Ch) IC_CLR_TX_OVER.....	181
REG 12-21 (Offset 0050h) IC_CLR_RD_REQ	181
REG 12-22 (Offset 0054h) IC_CLR_TX_ABRT	182
REG 12-23 (Offset 0058h) IC_CLR_RX_DONE.....	182

REG 12-24 (Offset 005Ch) IC_CLR_ACTIVITY	182
REG 12-25 (Offset 0060h) IC_CLR_STOP_DET	183
REG 12-26 (Offset 0064h) IC_CLR_START_DET	183
REG 12-27 (Offset 0068h) IC_CLR_GEN_CALL	183
REG 12-28 (Offset 006Ch) IC_ENABLE	184
REG 12-29 (Offset 0070h) IC_STATUS	184
REG 12-30 (Offset 0074h) IC_TXFLR	185
REG 12-31 (Offset 0078h) IC_RXFLR	185
REG 12-32 (Offset 007Ch) IC_SDA_HOLD	186
REG 12-33 (Offset 0080h) IC_TX_ABRT_SOURCE	186
REG 12-34 (Offset 0084h) IC_SLV_DATA_NACK_ONLY	188
REG 12-35 (Offset 0088h) IC_DMA_CR	188
REG 12-36 (Offset 008Ch) IC_DMA_TDLR	189
REG 12-37 (Offset 0090h) IC_DMA_RDLR	189
REG 12-38 (Offset 0094h) IC_SDA_SETUP	189
REG 12-39 (Offset 0098h) IC_ACK_GENERAL_CALL	190
REG 12-40 (Offset 009Ch) IC_ENABLE_STATUS	190
REG 12-41 (Offset 00A0h) IC_DMA_CMD	191
REG 12-42 (Offset 00A4h) IC_DMA_LEN	191
REG 12-43 (Offset 00A8h) IC_DMA_MOD	191
REG 12-44 (Offset 00ACh) IC_SLEEP	192
REG 12-45 (Offset 00B0h) IC_DEBUG_SEL	192
REG 12-46 (Offset 00B4h) IC_STRCH_DLY	192
REG 12-47 (Offset 00B8h) IC_ANA_FLTR_SDA_RL	193
REG 12-48 (Offset 00BCh) IC_ANA_FLTR_SDA_RM	193
REG 12-49 (Offset 00C0h) IC_ANA_FLTR_SCL_RL	193
REG 12-50 (Offset 00C4h) IC_ANA_FLTR_SCL_RM	194
REG 12-51 (Offset 00C8h) IC_ANA_FLTR_SDA_CL	194
REG 12-52 (Offset 00CCh) IC_ANA_FLTR_SDA_CM	194
REG 12-53 (Offset 00D0h) IC_ANA_FLTR_SCL_CL	195
REG 12-54 (Offset 00D4h) IC_ANA_FLTR_SCL_CM	195
REG 12-55 (Offset 00E8h) IC_CLR_DMA_I2C_DONE	195
REG 12-56 (Offset 00EcH) IC_FILTER	196
REG 12-57 (Offset 00F4h) IC_SAR1	196
REG 12-58 (Offset 00FCh) IC_COMP_VER	197
REG 13-1 DMA_SARx	206
REG 13-2 DMA_DARx	206
REG 13-3 DMA_CTLx	207

Table 13-4 CTLx.SRC_MSIZ/DEST_MSIZ Decoding.....	208
Table 13-5 CTLx.SRC_TR_WIDTH/DEST_TR_WIDTH Decoding.....	208
Table 13-6 CTLx.TT_FC Field Decoding	208
REG 13-7 DMA_CFGx.....	209
REG 13-8 (Offset 02C0h) DMA_RawTfr	210
REG 13-9 (Offset 02C8h) DMA_RawBlock	210
REG 13-10 (Offset 02D0) DMA_RawSrcTran	211
REG 13-11 (Offset 02D8h) DMA_RawDstTran	211
REG 13-12 (Offset 02E0h) DMA_RawErr	212
REG 13-13 (Offset 02E8h) DMA_StatusTfr	212
REG 13-14 (Offset 02F0h) DMA_StatusBlock	212
REG 13-15 (Offset 02F8h) DMA_StatusSrcTran	213
REG 13-16 (Offset 0300h) DMA_StatusDstTran	213
REG 13-17 (Offset 0308h) DMA_StatusErr.....	213
REG 13-18 (Offset 0310h) DMA_MaskTfr	214
REG 13-19 (Offset 0318h) DMA_MaskBlock	214
REG 13-20 (Offset 0320h) DMA_MaskSrcTran	215
REG 13-21 (Offset 0328h) DMA_MaskDstTran	216
REG 13-22 (Offset 0330h) DMA_MaskErr	216
REG 13-23 (Offset 0338h) DMA_ClearTfr.....	217
REG 13-24 (Offset 0340h) DMA_ClearBlock	217
REG 13-25 (Offset 0348h) DMA_ClearSrcTran	218
REG 13-26 (Offset 0350h) DMA_ClearDstTran.....	218
REG 13-27 (Offset 0358h) DMA_ClearErr	219
REG 13-28 (Offset 0360h) DMA_StatusInt	219
REG 13-29 (Offset 0398h) DMA_DmaCfgReg.....	219
REG 13-30 (Offset 03A0h) DMA_ChEnReg	220
REG 13-31 (Offset 03B8h) DMA_SoftResetReg	221
Table 14-1 SDIO Speed	223
Table 14-2 SDIO/SPI Pin Table	224
Table 14-3 SDIO Register Address	235
Table 14-4 TX Descriptor	237
Table 14-5 RX Descriptor	238
Table 14-6 SPI Slave Pins	239
Table 14-7 SPI Transfer Command Stage.....	240
Table 14-8 Format of Control Register and Data FIFO.....	243
Table 14-9 Command Fields	244
REG 14-10 (Offset 0000h) SDIO_TX_CTRL.....	245

REG 14-11 (Offset 0002h) SDIO_STATUS_RECOVERY_TIMEOUT	245
REG 14-12 (Offset 0014h) SDIO_HIMR.....	246
REG 14-13 (Offset 0018h) SDIO_HISR	246
REG 14-14 (Offset 001Ch) RX0_REQ_LEN	247
REG 14-15 (Offset 0020h) FREE_TXBD_NUM.....	248
REG 14-16 (Offset 0024h) TX_SEQNUM.....	248
REG 14-17 (Offset 0038h) HCPWM	248
REG 14-18 (Offset 003Ah) HCPWM2.....	249
REG 14-19 (Offset 0060h) REG_SDIO_H2C_MSG	249
REG 14-20 (Offset 0064h) REG_SDIO_C2H_MSG	249
REG 14-21 (Offset 0068h) REG_SDIO_H2C_MSG_EXT.....	250
REG 14-22 (Offset 006Ch) REG_SDIO_C2H_MSG_EXT.....	250
REG 14-23 (Offset 0080h) HRPWM	250
REG 14-24 (Offset 0082h) HRPWM2	251
REG 14-25 (Offset 0084h) HPS_CLKR.....	251
REG 14-26 (Offset 0087h) CPU_INDICATION.....	251
REG 14-27 (Offset 0088h) CMD_IN_TO_RSP_OUT_TIME	252
REG 14-28 (Offset 00C0h) ERR_FLAG	252
REG 14-29 (Offset 00C2h) SDIO_CMD_ERRCNT	252
REG 14-30 (Offset 00C3h) SDIO_DATA_ERRCNT	252
REG 14-31 (Offset 00C4h) SDIO_CRC_ERR_INDEX	253
REG 14-32 (Offset 00D0h) SDIO_AVAI_BD_NUM_TH_L	253
REG 14-33 (Offset 00D4h) SDIO_AVAI_BD_NUM_TH_H.....	253
REG 14-34 (Offset 00D8h) SDIO_RX_AGG_CFG	254
REG 14-35 (Offset 00DCh) SDIO_CMD_TIMING_CTRL.....	254
REG 14-36 (Offset 00DDh) SD_DATA01_TIMING_CTRL.....	254
REG 14-37 (Offset 00DEh) SD_DATA23_TIMING_CTRL	255
REG 14-38 (Offset 01A0h) SPDIO_TXBD_ADDR	255
REG 14-39 (Offset 01A4h) SPDIO_TXBD_NUM	255
REG 14-40 (Offset 01A8h) SPDIO_TXBD_WPTR	256
REG 14-41 (Offset 01ACh) SPDIO_TXBD_RPTR	256
REG 14-42 (Offset 01B0h) SPDIO_RXBD_ADDR	256
REG 14-43 (Offset 01B4h) SPDIO_RXBD_NUM	257
REG 14-44 (Offset 01B6h) SPDIO_RXBD_C2H_WPTR	257
REG 14-45 (Offset 01B8h) SPDIO_RXBD_C2H_RPTR.....	257
REG 14-46 (Offset 01BAh) HCI_RX_REQ.....	258
REG 14-47 (Offset 01BBh) CPU_RST_SDIO_DMA.....	258
REG 14-48 (Offset 01BCh) SDIO_RX_REQ_ADDR	258

REG 14-49 (Offset 01C0h) SPDIO_CPU_INT_MASK.....	259
REG 14-50 (Offset 01C2h) SPDIO_CPU_INT_REG.....	260
REG 14-51 (Offset 01C4h) CCPWM	261
REG 14-52 (Offset 01C5h) SYS_CPU_INDICATION	261
REG 14-53 (Offset 01C6h) CCPWM2	262
REG 14-54 (Offset 01C8h) REG_CPU_H2C_MSG	262
REG 14-55 (Offset 01CCh) REG_CPU_C2H_MSG	262
REG 14-56 (Offset 01D0h) CRPWM	263
REG 14-57 (Offset 01D2h) CRPWM2	263
REG 14-58 (Offset 01D4h) AHB_DMA_CTRL	264
REG 14-59 (Offset 01D8h) FREE_RXBD_COUNT.....	265
REG 14-60 (Offset 01DCh) REG_CPU_H2C_MSG_EXT	265
REG 14-61 (Offset 01E0h) REG_CPU_C2H_MSG_EXT	265
REG 14-62 (Offset 004h) SPI_INT	266
REG 14-63 (Offset 014h) SPI_HIMR.....	266
REG 14-64 (Offset 018h) SPI_HISR	267
REG 14-65 (Offset 01Ch) RX0_REQ_LEN	267
REG 14-66 (Offset 020h) FREE_TX_BD_NUM.....	268
REG 14-67 (Offset 024h) TX_SEQNUM.....	268
REG 14-68 (Offset 038h) HCPWM	268
REG 14-69 (Offset 03Ah) HCPWM2.....	269
REG 14-70 (Offset 040h) SPI_AVAI_PGTH_L.....	269
REG 14-71 (Offset 044h) SPI_AVAI_PGTH_H.....	269
REG 14-72 (Offset 048h) SPI_RX_AGG	270
REG 14-73 (Offset 04Ch) REG_SPI_H2C_MSG.....	270
REG 14-74 (Offset 050h) REG_SPI_C2H_MSG	270
REG 14-75 (Offset 080h) HRPWM	271
REG 14-76 (Offset 084h) HPS_CLKR	271
REG 14-77 (Offset 087h) CPU_INDICATION.....	271
REG 14-78 (Offset 088h) 32K_TRANSPARENT	272
REG 14-79 (Offset 08Ah) 32K_IDLE	272
REG 14-80 (Offset 08Ch) DELY_LINE_SEL	272
REG 14-81 (Offset 0F0h) SPI_CFG.....	273
REG 15-1 (Offset 0000h) SPIC_CTRLR0.....	284
REG 15-2 (Offset 0004h) SPIC_CTRLR1.....	285
REG 15-3 (Offset 0008h) SPIC_SSIENR	285
REG 15-4 (Offset 0010) SPIC_SER	286
REG 15-5 (Offset 0014) SPIC_BAUDR	286

REG 15-6 (Offset 0020h) SPIC_TXFLR.....	286
REG 15-7 (Offset 0024h) SPIC_RXFLR.....	287
REG 15-8 (Offset 0028h) SPIC_SR.....	287
REG 15-9 (Offset 0058h) SPIC_IDR	288
REG 15-10 (Offset 005Ch) SPIC_VERSION	288
REG 15-11 (Offset 0060h) SPIC_DR	288
REG 15-12 (Offset 00E0h) SPIC_READ_FAST_SINGLE.....	289
REG 15-13 (Offset 00E4h) SPIC_READ_DUAL_DATA.....	289
REG 15-14 (Offset 00E8h) SPIC_READ_DUAL_ADDR_DATA	289
REG 15-15 (Offset 00Ec h) SPIC_READ_QUAD_DATA.....	290
REG 15-16 (Offset 00F0h) SPIC_READ_QUAD_ADDR_DATA.....	290
REG 15-17 (Offset 00F4h) SPIC_WRITE_SINGLE.....	291
REG 15-18 (Offset 0108h) SPIC_WRITE_ENABLE	291
REG 15-19 (Offset 010Ch) SPIC_READ_STATUS.....	291
REG 15-20 (Offset 0110h) SPIC_CTRLR2.....	292
REG 15-21 (Offset 0114h) SPIC_FBAUDR	292
REG 15-22 (Offset 0118h) SPIC_ADDR_LENGTH.....	293
REG 15-23 (Offset 011C) SPIC_AUTO_LENGTH	293
REG 15-24 (Offset 0120) SPIC_VALID_CMD	295
REG 15-25 (Offset 0128h) SPIC_FLUSH_FIFO	295
REG 16-1 (Offset 0000h) GPIO_IT_STS	300
REG 16-2 (Offset 0004h) GPIO_EI_EN	300
REG 16-3 (Offset 0008h) GPIO_LI_EN	301
REG 16-4 (Offset 000Ch) GPIO_IP_STS	301
REG 16-5 (Offset 0014h) GPIO_IR_EN	303
REG 16-6 (Offset 0018h) GPIO_IF_EN	303
REG 16-7 (Offset 001Ch) GPIO_ID_EN	304
REG 16-8 (Offset 0020h) GPIO_IM_STS	304
REG 16-9 (Offset 0024h) GPIO_IM_EN	304
REG 16-10 (Offset 0028h) GPIO_IM_DIS	305
REG 16-11 (Offset 002Ch) GPIO_RAW_INT_STS	305
REG 16-12 (Offset 0030h) GPIO_INT_STS	305
REG 16-13 (Offset 0034h) GPIO_INT_CLR	306
REG 16-14 (Offset 0038h) GPIO_INT_EN_STS	306
REG 16-15 (Offset 003Ch) GPIO_INT_EN	306
REG 16-16 (Offset 0040h) GPIO_INT_DIS	307
REG 16-17 (Offset 0050h) GPIO_INT0_SEL	307
REG 16-18 (Offset 0054h) GPIO_INT1_SEL	307

REG 16-19 (Offset 0058h) GPIO_INT2_SEL	308
REG 16-20 (Offset 005Ch) GPIO_INT2_SEL	308
REG 16-21 (Offset 0060) GPIO_INT4_SEL.....	309
REG 16-22 (Offset 0064) GPIO_INT5_SEL.....	309
REG 16-23 (Offset 0068) GPIO_INT6_SEL.....	309
REG 16-24 (Offset 006C) GPIO_INT4_SEL	310
REG 16-25 (Offset 0070) GPIO_INT4_SEL.....	310
REG 16-26 (Offset 0074) GPIO_INT9_SEL.....	311
REG 16-27 (Offset 0078) GPIO_INTA_SEL.....	311
REG 16-28 (Offset 007C) GPIO_INTB_SEL	312
REG 16-29 (Offset 0080) GPIO_INTC_SEL.....	312
REG 16-30 (Offset 0084) GPIO_INTD_SEL	312
REG 16-31 (Offset 0088) GPIO_INTE_SEL.....	313
REG 16-32 (Offset 008C) GPIO_INTF_SEL.....	313
REG 16-33 (Offset 00F0) GPIO_DEB_STS.....	314
REG 16-34 (Offset 00F4) GPIO_DEB_EN.....	314
REG 16-35 (Offset 00F8) GPIO_DEB_DIS.....	314
REG 16-36 (Offset 00FC) DEB_DP_STS	315
REG 16-37 (Offset 0100) GPIO_DEB0_SEL.....	315
REG 16-38 (Offset 0104) GPIO_DEB1_SEL.....	315
REG 16-39 (Offset 0108) GPIO_DEB2_SEL.....	316
REG 16-40 (Offset 010C) GPIO_DEB3_SEL	316
REG 16-41 (Offset 0110) GPIO_DEB4_SEL.....	317
REG 16-42 (Offset 0114) GPIO_DEB5_SEL.....	317
REG 16-43 (Offset 0118) GPIO_DEB6_SEL.....	317
REG 16-44 (Offset 011C) GPIO_DEB7_SEL	318
REG 16-45 (Offset 0120) GPIO_DEB8_SEL.....	318
REG 16-46 (Offset 0124) GPIO_DEB9_SEL.....	319
REG 16-47 (Offset 0128) GPIO_DEBA_SEL	319
REG 16-48 (Offset 012C) GPIO_DEBB_SEL	319
REG 16-49 (Offset 0130) GPIO_DEBC_SEL	320
REG 16-50 (Offset 0134) GPIO_DEBD_SEL	320
REG 16-51 (Offset 0138) GPIO_DEBE_SEL.....	321
REG 16-52 (Offset 013C) GPIO_DEBF_SEL.....	321
REG 16-53 (Offset 0200) GPIOA_DMD_STS	321
REG 16-54 (Offset 0204) GPIOA_IDM_EN	322
REG 16-55 (Offset 0208) GPIOA_ODM_EN	322
REG 16-56 (Offset 020C) GPIOA_OD_STS.....	322

REG 16-57 (Offset 0210) GPIOA_ODL_EN	323
REG 16-58 (Offset 0214) GPIOA_ODH_EN	323
REG 16-59 (Offset 0218) GPIOA_ODT_EN	323
REG 16-60 (Offset 021C) GPIOA_DP_STS	324
Table 18-1 SCD_DT1_Format.....	336
Table 18-2 SCD_DT1_Description.....	336
Table 18-3 SCD_CS_Format	337
Table 18-4 SCD_CS_Description	338
Table 18-5 SCD_DT2_Format.....	342
Table 18-6 SCD_DT2_Description.....	342
Table 18-7 DCD_DT1_Format	343
Table 18-8 DCD_DT1_Description	343
Table 18-9 DCD_DT2_Format	343
Table 18-10 DCD_DT2_Description	344
REG 18-11 (Offset 0000h) SDSR	344
REG 18-12 (Offset 0004h) SDFWR.....	345
REG 18-13 (Offset 0008h) SDSWR	346
REG 18-14 (Offset 0010h) IPCSR_CONF_INT	346
REG 18-15 (Offset 0014h) IPCSR_INT_MASK	347
REG 18-16 (Offset 0018h) IPCSR_DBG_DMA	348
REG 18-17 (Offset 001Ch) IPCSR_ERR_STATUS.....	349
REG 18-18 (Offset 0020h) IPCSR_SUM_OF_AADL.....	351
REG 18-19 (Offset 0024h) IPCSR_SUM_OF_EDL	351
REG 18-20 (Offset 0028h) IPCSR_SUM_OF_APDL	352
REG 18-21 (Offset 002Ch) IC_HS_SCL_LCNT	352
REG 18-22 (Offset 0030h) IPCSR_SWP_BURST.....	353
REG 18-23 (Offset 1000h) DDSR.....	353
REG 18-24 (Offset 1004h) DDFWR	354
REG 18-25 (Offset 1008h) DDSWR	355
REG 18-26 (Offset 100Ch) IPCSR_CONF_PKTABR	355
REG 18-27 (Offset 1010h) IPCSR_DBG_SDDR.....	355
REG 18-28 (Offset 1014h) IPCSR_DBG_DDDR	356
Table 20-1 Supported BT Features	364

List of Figures

Figure 1-1 Ameba-Z II Block Diagram	37
Figure 1-2 Ameba-Z II Power Architecture	39
Figure 2-1 Flash Memory Layout.....	41
Figure 2-2 eFuse Memory For Users	42
Figure 2-3 eFuse Block Diagram	43
Figure 3-1 Application Scenario Flow.....	44
Figure 3-2 Power Mode Block Diagram.....	46
Figure 3-3 Shutdown Mode.....	50
Figure 3-4 Deep Sleep Mode.....	51
Figure 3-5 Standby Mode	52
Figure 3-6 Sleep Mode	54
Figure 3-7 Snooze Mode.....	55
Figure 4-1 Ameba-Z II Secure Boot process with multiple stages.....	60
Figure 4-2 The Debugger Protection Mechanism Architecture.	62
Figure 4-3 The Debugger State Transition.....	63
Figure 4-4 Ameba-Z II Device Lifecycle State	64
Figure 6-1 The System of CM33	70
Figure 6-2 The System of Real-M300	71
Figure 7-1 HS GTimer Function Block.....	80
Figure 7-2 HS GTimer Architecture	80
Figure 7-3 HS GTimer Interrupt Event Diagram	82
Figure 7-4 LP GTimer Function Block	89
Figure 7-5 LP GTimer Architecture	89
Figure 7-6 LP GTimer Interrupt Event Diagram	91
Figure 8-1 Watchdog Timer Architecture.....	98
Figure 9-1 PWM Architecture.....	101
Figure 9-2 PWM Waveforms (PWMx_PERIOD = 9)	102
Figure 9-3 PWM Auto-Adjustment.....	103
Figure 9-4 PWM Interrupt Event Diagram (PWMx_PERIOD = 9)	104
Figure 10-1 UART Block Diagram.....	112
Figure 10-2 UART Wire Connection.....	113
Figure 10-3 UART Data Format.....	114
Figure 10-4 Hardware Flow Control Between 2 UARTs	116
Figure 10-5 UART Rx Filter Block Diagram.....	117
Figure 11-1 SPI in A Complete System	140

Figure 11-2 SPI Block Diagram	141
Figure 11-3 SPI Pin Connection	142
Figure 11-4 SPI Protocol: Mode 0	144
Figure 11-5 SPI Mode Protocol: Mode 1	144
Figure 11-6 SPI Mode Protocol: Mode 2	144
Figure 11-7 SPI Mode Protocol: Mode 3	144
Figure 11-8 SSI Protocol	145
Figure 12-1 I2C Block Diagram	162
Figure 12-2 I2C Master/Slave and Transmitter/Receiver Relationship	164
Figure 12-3 Data Transfer on The I2C Bus	165
Figure 12-4 START and STOP Conditions	167
Figure 12-5 7-Bit Address Format	167
Figure 12-6 10-Bit Address Format	168
Figure 12-7 Master-Transmitter Protocol	168
Figure 12-8 Master-Receiver (Slave-Transmitter) Protocol	169
Figure 12-9 IC_DATA_CMD Register Content	169
Figure 12-10 Master Transmitter – TX FIFO Empty and STOP Generation	170
Figure 12-11 Master Receiver – Tx FIFO Empty and STOP Generation	170
Figure 12-12 Master Transmitter – RESTART Generation	170
Figure 12-13 Master Receiver – RESTART Generation	170
Figure 12-14 Master Transmitter – STOP bit Set and Tx FIFO Not Empty	171
Figure 12-15 Master Receiver – STOP bit Set and Tx FIFO Not Empty	171
Figure 12-16 I2C 8-bit FIFO Content with Transfer Control Register	172
Figure 13-1 RTK-DMAC in A complete system	199
Figure 13-2 RTK-DMAC Block Diagram	200
Figure 13-3 DMA Hardware Handshake Signals	202
Figure 13-4 DMA Handshaking Loop	203
Figure 14-1 Ameba-Z II Works as An SDIO Device In A System	222
Figure 14-2 Ameba-Z II Works as An SPI Slave Device In A System	223
Figure 14-3 SDIO Device/SPI Slave Architecture	225
Figure 14-4 SDIO Device Interface Architecture	226
Figure 14-5 SPI Slave Interface Architecture	227
Figure 14-6 Software Architecture Example of Ameba-Z II SDIO NIC	228
Figure 14-7 Software Architecture Example of User Defined Protocol over SDIO/SPI Interface	228
Figure 14-8 SPDIO Tx DMA Configuration	229
Figure 14-9 SPDIO Tx DMA Behavior	231
Figure 14-10 SPDIO RxBD Format	233

Figure 14-11 SPDIO Rx DMA Configuration.....	233
Figure 14-12 SPDIO Rx DMA Behavior	234
Figure 14-13 Packet Format	236
Figure 14-14 Format for Packets Aggregation.....	236
Figure 14-15 The Connection of Ameba-Z II SPI Slave with Host System	239
Figure 14-16 The Timing Waveform for Ameba-Z II SPI Slave Interface	240
Figure 14-17 The SPI Slave Interface Protocol for Register Reading	241
Figure 14-18 The SPI Slave Interface Protocol for Register Writing	242
Figure 14-19 The SPI Slave Interface Protocol for Packet Rx (RX_FIFO reading).....	242
Figure 14-20 The SPI Slave Interface Protocol for Packet Tx (TX_FIFO writing)	243
Figure 14-21 The SPI Command Format	243
Figure 14-22 The SPI Status Format	244
Figure 15-1 Ameba-Z II SPIC in A Complete System	275
Figure 15-2 SPIC Block Diagram	276
Figure 15-3 Single Channel Connection (for One IO Mode).....	277
Figure 15-4 Multi-Channel Connection	278
Figure 15-5 SPI Protocol: Mode 0.....	278
Figure 15-6 SPI Mode Protocol : Mode 3	279
Figure 15-7 General Format of Transfer Mode.....	279
Figure 15-8 Transmit Mode: Type 1	280
Figure 15-9 Transmit Mode: Type 2	280
Figure 15-10 Transmit Mode: Type 3	280
Figure 15-11 Receive Mode.....	281
Figure 16-1 GPIO Architecture	297
Figure 16-2 The Correspond GPIO Interrupts	299
Figure 16-3 GPIO Debounce Operation Diagram	299
Figure 17-1 The Architecture of “Secure Code Engine”	326
Figure 17-2 The Flow of Generating Secure Image	326
Figure 17-3 The Expression of The Proper Noun for “Secure Code Engine”	327
Figure 17-4 The Process of CPU Fetching Instructions by Secure Code Engine	328
Figure 17-5 The Method of Remap for Secure Code Engine.....	328
Figure 18-1 Crypto Engine Block Diagram.....	331
Figure 18-2 Source Descriptor FIFO.....	333
Figure 18-3 Destination Descriptor FIFO	334
Figure 18-4 Key Array Element	334
Figure 18-5 IV Array Element	335
Figure 18-6 HMAC Key PAD Array Element	335
Sequential Hash references Figure 18-7	338

When programmer sets sequential hash message buffer, it always needs to set “1” to this bit.	
Sequential Hash references Figure 18-8.	338
Sequential Hash references Figure 18-9.	338
Sequential Hash references Figure 18-10	338
Figure 18-11 Sequential Hash Mechanism	341
Figure 19-1 Wi-Fi Block Diagram	358
Figure 19-2 Wi-Fi MAC TRX Flow	360
Figure 19-3 The State Machine of The Power Save Mode	361
Figure 19-4 The Behavior of The Power Save Mode	362
Figure 19-5 The Behavior of The Wake On Wireless LAN	363
Figure 20-1 Bluetooth Profiles	365
Figure 20-2 GATT Based Profile Hierarchy	366
Figure B-1 ARM ACK Certificate	370

History

Version	Date	Author	Change Note
1.0	2019/7/25	Scott C.	First release
1.1	2019/08/08	JD	Update I2C 10-bit
1.2	2019/9/9	Scott C.	Update SDIO device/SPI slave control registers

1 Product Overview

1.1 General Description

Realtek Ameba-Z II series are highly integrated single-chip low power 802.11n Wireless LAN (WLAN) network controllers. It combines a Real-M300 (also named KM4) MCU, Wi-Fi MAC, a 1T1R capable Wi-Fi baseband, RF, and Bluetooth in a single chip. It also provides a bunch of configurable GPIOs which are configured as digital peripherals for different applications and control usage.

The Real-M300(KM4) MCU which is ARMv8m architecture provides significant performance improvement, low power consumption, enhanced debug features, DSP instruction, etc. Real-M300 could execute at the clock up to 100 MHz with 32K bytes instruction cache therefore the computing power should be sufficient for any kinds of IOT software.

For wireless connectivity, Ameba-Z II supports Wi-Fi (802.11n) and Bluetooth Smart Ready (Bluetooth Low Energy) full protocol stack in the chip. The Wi-Fi of Ameba-Z II provides complete 802.11n solution for 2.4 GHz band with 65Mbps receive PHY rate and 65Mbps transmit PHY rate using 20 MHz bandwidth.

Ameba-Z II series also integrate internal memories for complete Wi-Fi protocol functions. The embedded memory configuration also provides simple application developments. With the security unit such as Crypto engine and eFuse, the protection for software applications could be achieved.

Plenty of peripherals in Ameba-Z II make IOT system design much easier than other solutions. General purpose timer, PWM, UART, SPI, I2C, SDIO and GPIO in this chip could be multiplexed according to the external connection requirement.

Related Documents

- *Armv8-M Architecture Reference Manual*. Available from <https://developer.arm.com>

1.2 System Architecture

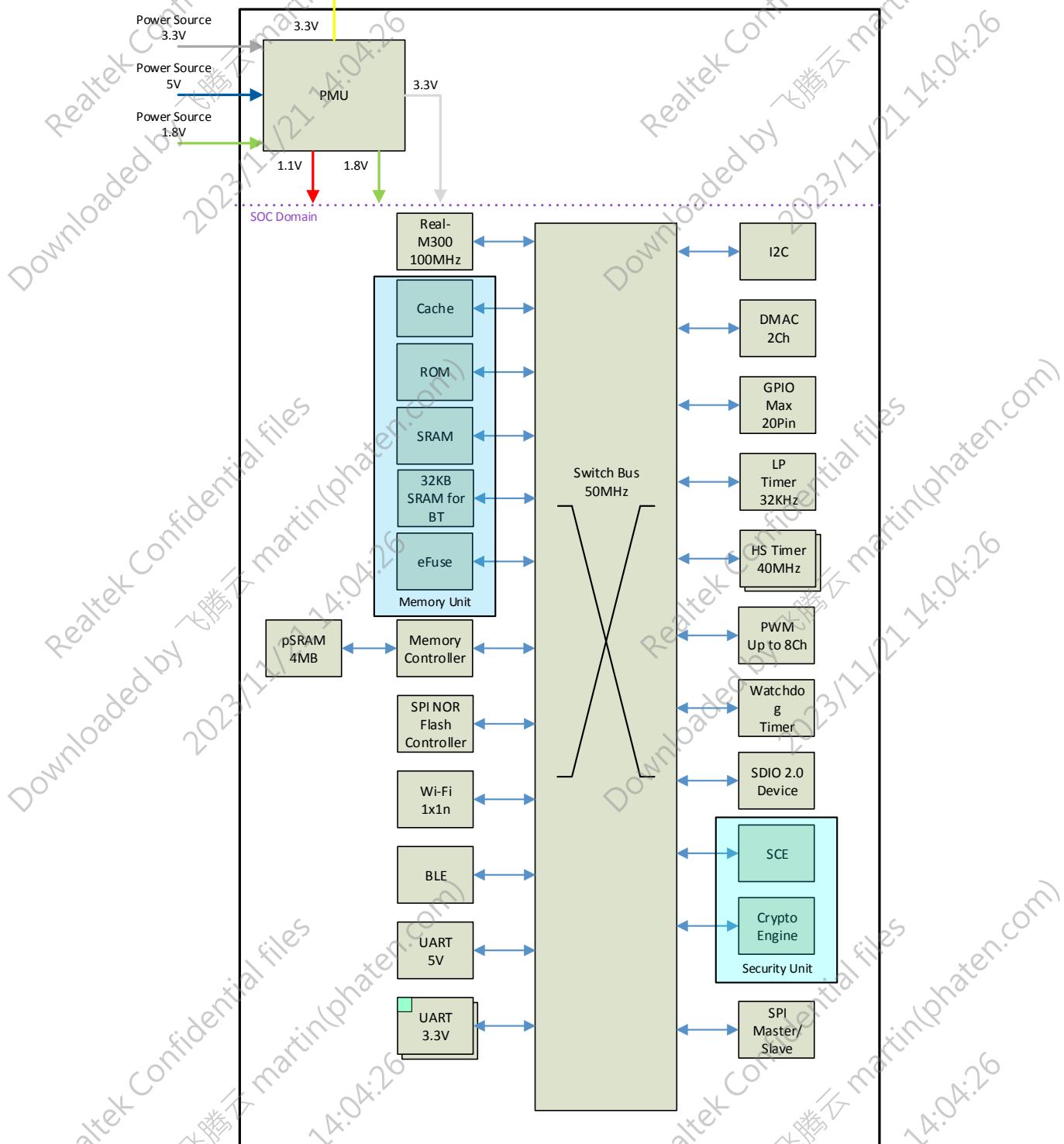


Figure 1-1 Ameba-Z II Block Diagram

1.3 Security Architecture

The section gives a view of Ameba-Z II platform security architecture. Ameba-Z II platform security implementation is based on the integration of following components:

- Secure boot verifies the authenticity and integrity of the firmware images which will be loaded.
- Protection for keys and system configuration which are stored in eFuse.
- A lifecycle management for the control of debug port, IC test port and in-system flash programming function.
- Crypto Engine hardware for real-time data encryption/decryption/hash-authentication. It accelerates the firmware image verification and decryption at boot time and the data encryption/decryption of secure network communication.
- Flash memory protection
- Debugger protection

1.4 Power Architecture

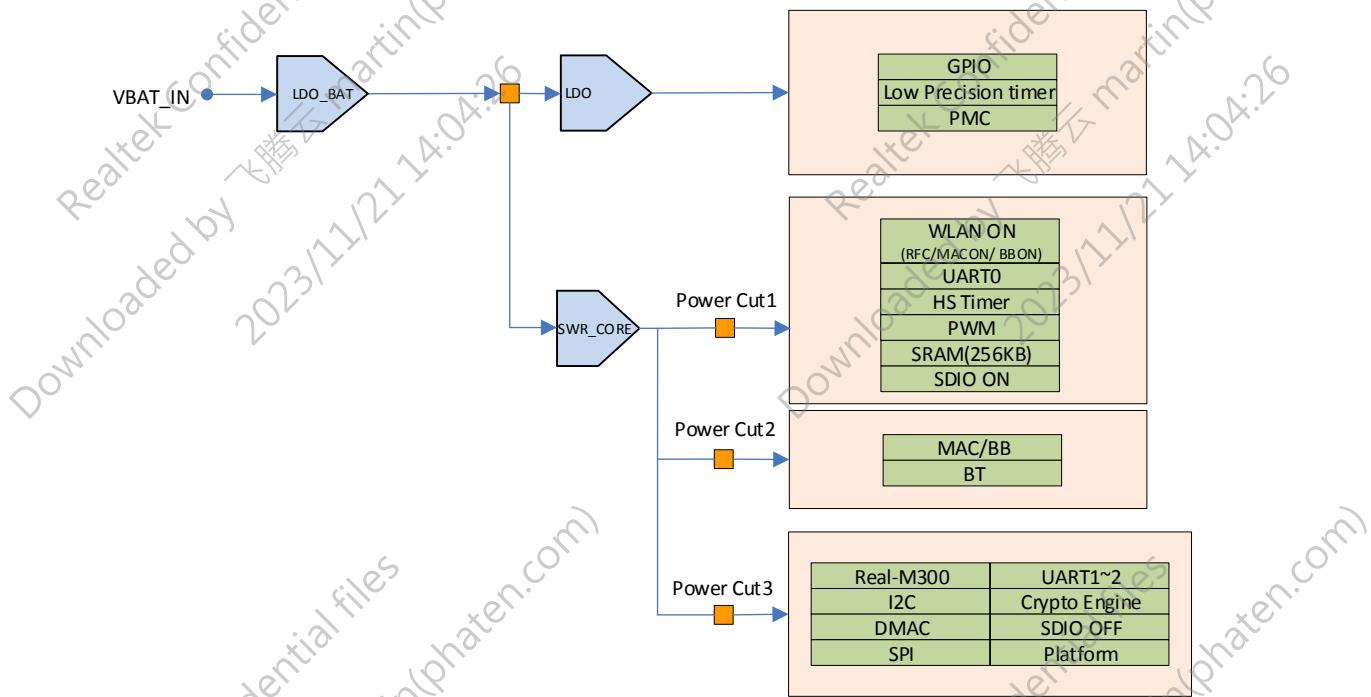


Figure 1-2 Ameba-Z II Power Architecture

Figure 1-2 shows the Ameba-Z II power architecture, it has two LDO (Low Dropout Regulator) and one SWR (Switching Regulator). The input power of the LDO_BAT was from VBAT_IN and supplies all power to LDO and SWR_CORE. The GPIO, low precision timer, and PMC are powered by LDO. The SWR_CORE supplies power to three parts, each part will be described below:

- Power Cut1 included WLAN ON, UART0, HS Timer, PWM, SRAM and SDIO ON.
- Power Cut2 included MAC/BB and BT.
- Power Cut3 included Real-M300, I2C, DMAC, SPI, UART1~2, Crypto engine, SDIO OFF and Platform.

Note: The WLAN module is powered by SWR_CORE. If WLAN module wants to continue its work, SWR_CORE can't be cut off when the chip enters low power mode.

2 Memory Organization

2.1 Overview

The program memory, data memory, registers, and I/O ports in Ameba-Z II are organized within the same 4G Byte address space. The byte order in memory is little endian format. Table 2-1 shows the address mapping available in Ameba-Z II.

Table 2-1 Address Mapping

IP Function	Address	Size	Cache
ITCM (ROM)	0x00000000 – 0x0005FFFF	384 KB	X
TCM SRAM	0x10000000 – 0x1003FFFF	256 KB	X
Additional SRAM for BT	0x20000000 – 0x20007FFF	32 KB	O
SPI Flash Controller	0x40020000 – 0x40020FFF	4 KB	X
SYS Control	0x40000000 – 0x400007FF	2 KB	X
GPIO	0x40001000 – 0x400017FF	2 KB	X
PWM	0x40001C00 – 0x40001FFF	1 KB	X
HS Timer	0x40002000 – 0x40002FFF	4 KB	X
UART0	0x40003000 – 0x400033FF	1 KB	X
LP Timer	0x40003800 – 0x40003FFF	2 KB	X
UART1	0x40040000 – 0x400403FF	1 KB	X
UART2	0x40040400 – 0x400407FF	1 KB	X
SPI	0x40042000 – 0x400423FF	1 KB	X
I2C	0x40044000 – 0x400443FF	1 KB	X
SDIO 2.0 Device	0x40050000 – 0x40053FFF	16 KB	X
DMAC	0x40060000 – 0x400607FF	2 KB	X
Crypto Engine	0x40070000 – 0x40073FFF	16 KB	X
WLAN	0x40080000 – 0x400BFFFF	256 KB	X
pSRAM Controller	0x40600000 – 0x40600FFF	4 KB	X
pSRAM	0x60000000 – 0x603FFFFF	4 MB	O
SPI NOR Flash Memory	0x98000000 – 0x9BFFFFFF	64 MB	O

2.2 Internal ROM

384 KB ROM is integrated to provide high access speed, low leakage memory. The ROM lib provides the following functions:

- Boot code and MCU initialization
- Non-flash booting functions and drivers
- System level control (interrupt/misc.)
- Utility (memory/shell command, etc.)

2.3 Real-M300 Embedded SRAM

256 KB SRAM is integrated to provide data and buffer usage and can be accessed as byte (8 bits), half-word (16 bits) or one word (32 bits).

2.4 Retention Memory

The embedded 256 KB SRAM can be used as retention memory, in order to save data with minimal power during sleep mode.

2.5 SPI-NOR Flash Memory

The flash memory layout is shown in Figure 2-1.

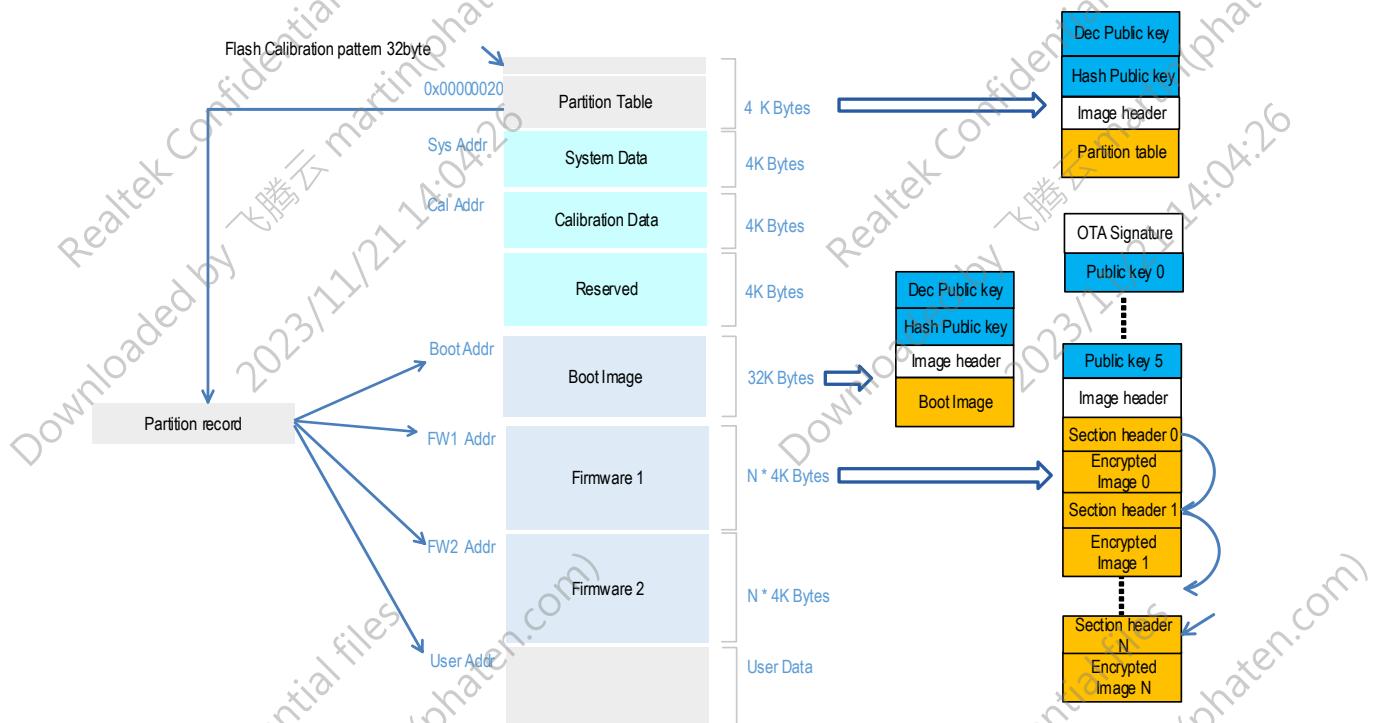


Figure 2-1 Flash Memory Layout

Table 2-2 shows the description of flash memory layout.

Table 2-2 Description of Flash Memory Layout

Item	Address	Address adjustable	Size	Description
Partition Table	0x00000000 – 0x00000FFF	N	4 KB	The first 32 Bytes is flash calibration pattern. The actual partition table starts from 0x20
System Data	0x00001000 – 0x00001FFF	N	4 KB	To store some system settings
Calibration Data	0x00002000 – 0x00002FFF	N	4 KB	Reserved. User don't need to configure this portion
Reserved	0x00003000 – 0x00003FFF	N	4 KB	Reserved. User don't need to configure this portion
Boot Image	0x00004000 – 0x0000BFFF	Y	32 KB	Bootloader code/data
Firmware 1	0x0000C000 –	Y	N*4 KB	Application image

2.6 pSRAM

For RTL8720CM and RTL8710CM series of packages, a 4 MB x8 IO DDR pSRAM is included.

2.7 eFuse

The eFuse belongs to One Time Programmable (OTP) technology. Its default value is '1', so a default eFuse byte data is '0xFF'. It can be programmed from '1' to '0' only once. This means that it can not be reverted back from '0' to '1'. eFuse can be used to hold the individual and stable data such as key, calibration data, MAC address, specific setting, etc.

The Ameba-Z II eFuse provides 32bytes Normal Data eFuse memory and 32 bytes Secret Data eFuse memory for users. User can read/write each byte of them by eFuse APIs.

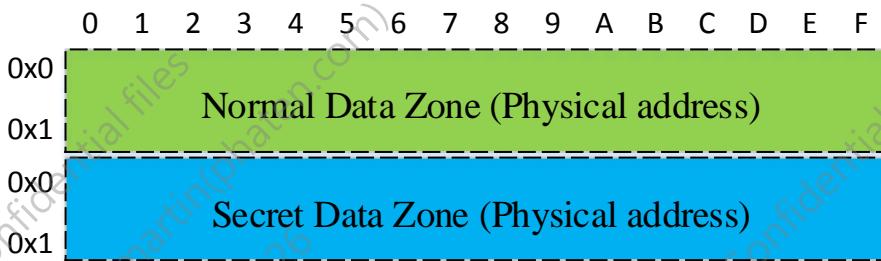


Figure 2-2 eFuse Memory For Users

eFuse write-APIs help users to set write signal, physical address, write data to control registers. eFuse Top will program the desired address of eFuse to become the write data. eFuse read-APIs also help users to set read signal, physical address, and eFuse Top will fill the data of

desired eFuse address to corresponding control register.

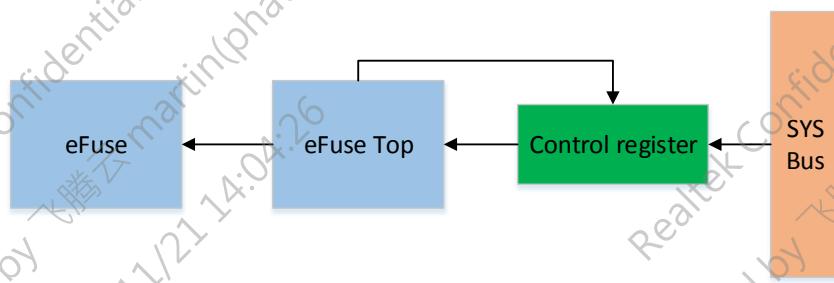


Figure 2-3 eFuse Block Diagram

3 Power Mode and Control

3.1 Overview

3.1.1 Application Scenario

Ameba-Z II achieves low power consumption with a combination of several proprietary technologies. The power-saving architecture features six reduced power modes of operation: active, snooze, sleep, standby, deepsleep, shutdown mode. With the elaborate architecture, the battery life of whole IOT system could be extended.

For reading pen application, it can divide into three-scenario. First, press the power button to power on reading pen to active mode and then connect to the cloud to download data. Second, once the reading pen without any activity for 2 minutes, the system will go to sleep mode (for system fast resume and keep WIFI connect) or standby mode (for lower power consumption and keep WIFI connect) and regularly wake up to receive WLAN beacon while into snooze mode. At last, without using the reading pen exceeds 10 minutes, the system will into deep sleep mode and waiting for any button signals to wake up system into active mode. The application scenario flow was shown in Figure 3-2

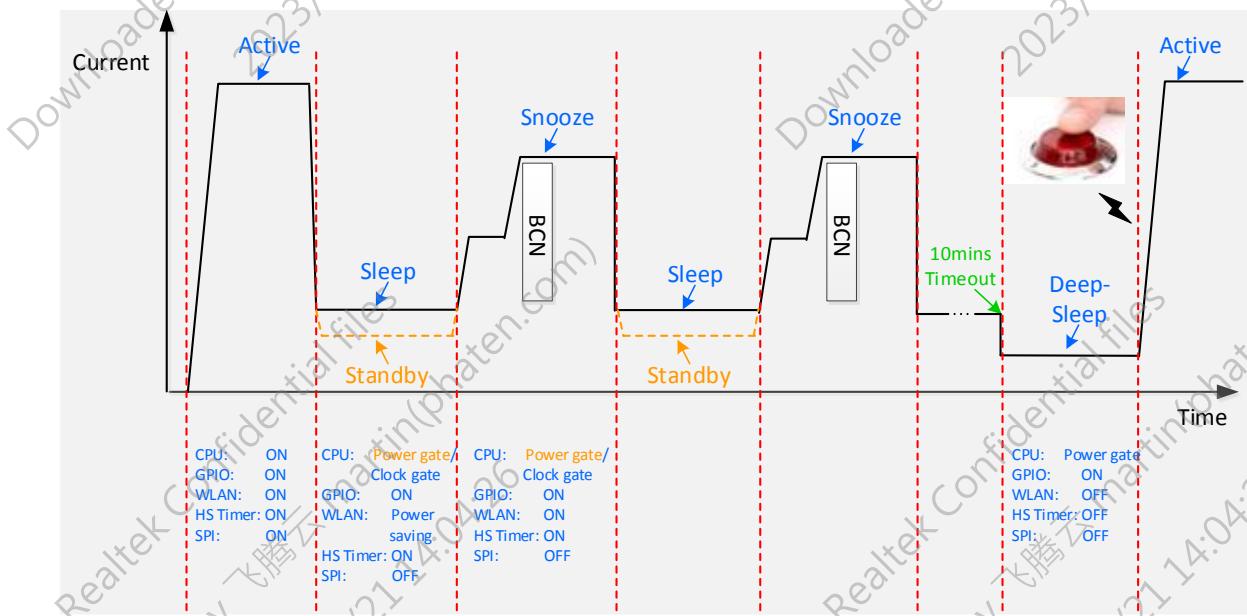


Figure 3-1 Application Scenario Flow

3.1.2 Features

- **Active Mode:** The Real-M300 in active mode and all peripherals are available.
- **Snooze Mode:** In this mode system can regularly wake up to receive WLAN beacon without software intervention. The significant difference between Snooze and Sleep/Standby mode is WLAN capability and could be receive and transmit beacon in this state.
- **Sleep Mode:** The Real-M300 in clock-gated and can be woken up by most of peripherals. The system resume time could be much faster than the Standby mode and the WLAN could be ON or power saving mode in this state.
- **Standby Mode:** The Real-M300 in power-gated and can be woken up by most of peripherals. The power consumption could be lower than the Sleep mode and the WLAN could be ON or power saving mode in this state.
- **DeepSleep Mode:** The lowest power consumption than the other power mode except for shutdown mode, it can be only woken up by LP Timer or GPIO.
- **Shutdown Mode:** The Real-M300 will be shutdown while CHIP_EN was Low.

3.1.3 Power Mode and Power Consumption

The mode changing diagram is given in Figure 3-2.

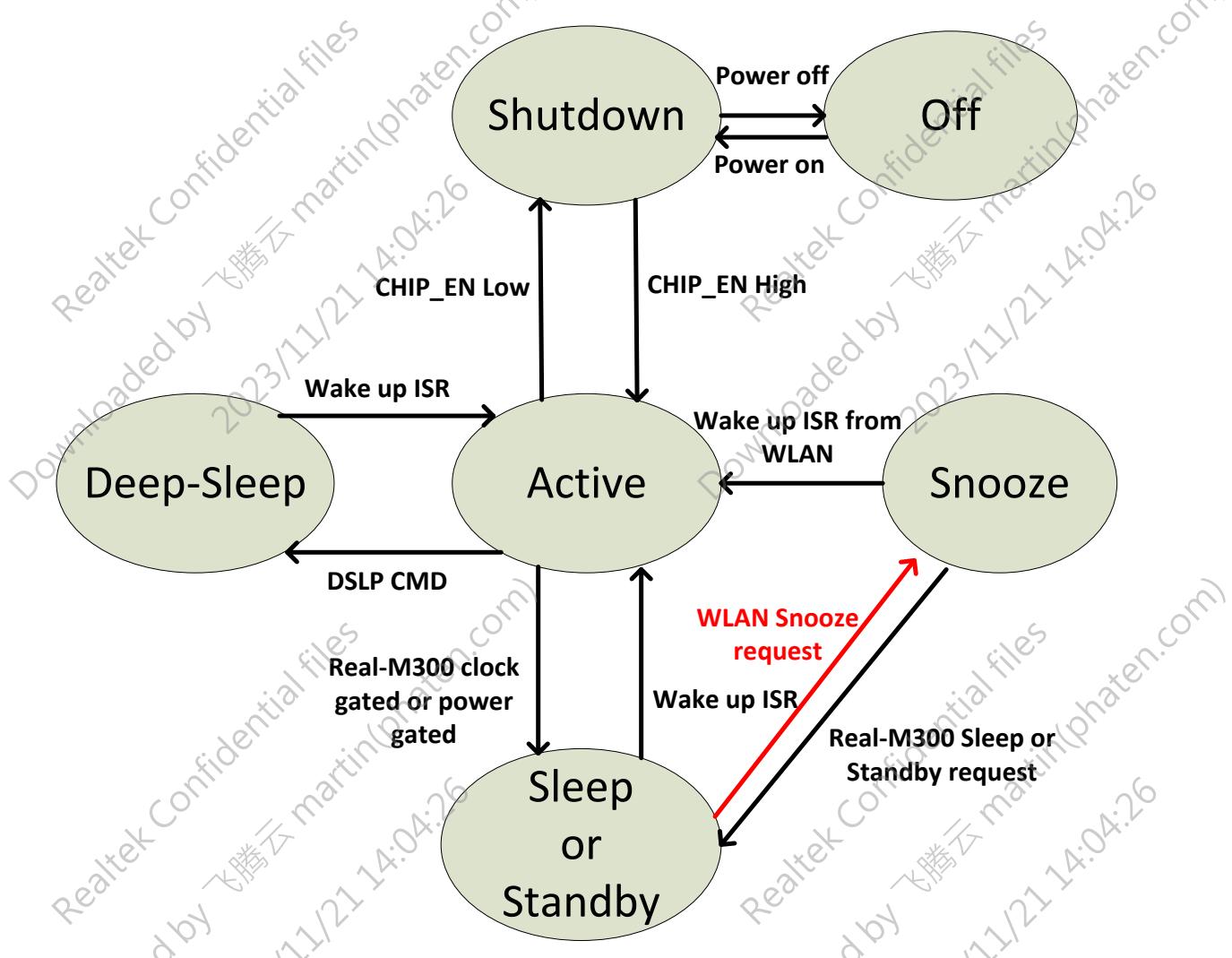


Figure 3-2 Power Mode Block Diagram

In Figure 3-2, the power mode can be divided into 6 states except for "off" state and the each power consumption was shown in Table 3-1. The introduction of each power mode, clock-gated and power-gated state will be in the following sections. Clock/power gated state could be regarded as a status of any hardware.

3.1.3.1 Overview of Power Modes

- The active mode is the most important role in the whole system. User can invoke software command to make the system into deep sleep, sleep or standby mode while in active mode.
- The Snooze mode is the particular mode for WLAN function. In this mode WLAN hardware can regularly wake up to trigger PMC and PMC would make the whole system change power mode between Sleep/Standby and Snooze without software intervention.
- The Sleep mode is CPU and clock of peripherals in platform domain into clock-gated state.

Cache and SRAM can be set to enter low power state which could reduce power consumption. Only specified peripheral interrupts can wake CPU up and turn to active state from clock-gated state. The whole system could quickly restore to the previous working state without any reboot flow.

- In the Standby mode, CPU and all platform peripherals enter into power-gated state. Cache and SRAM can be set to enter low power state just like Sleep mode. Therefore, once specified interrupts wake CPU up, the system will force into fast boot flow since CPU would be like power on again.
- The deep sleep mode is with the lowest power consumption than the other power modes except for shutdown mode. All power of clocks, platform hardware and peripherals would be cut down. The SRAM is powered off and all memory content could be lost. Therefore, once specified interrupts wake CPU up, the system will execute the whole system reboot flow.
- At CHIP_EN is low state, the shutdown mode was all power is off, except 5V convert to 3.3V LDO. Once CHIP_EN is high the system will force into active mode.

3.1.3.2 Clock-gated

When clock-gated state is entered, the clock to the core and related hardware domain stops. Resumption from this state doesn't need any special sequence but restart the clock to the CPU core and related hardware. In clock-gated state, instruction execution of CPU is suspended until either a reset or an interrupt occurs. Peripheral functions could continue operating during clock-gated state and could generate interrupts to cause CPU to resume execution. Clock-gated state eliminates dynamic power used by the processor itself, memory systems (optional), related controllers, and internal buses. Wakeup from clock-gated state occurs whenever any event interrupt takes place and is sent to PMC.

3.1.3.3 Power-gated

When the hardware enters the power-gated state, the power of CPU and related hardware would be cut off. The high speed clock source would also be disabled since the related hardware have stop operation. The power-gated state can be terminated and go back to normal state by either a reset or certain specific interrupts sent to PMC. This state reduces hardware power consumption to a very low value.

Table 3-1 gives a power consumption (not include MCM pRAM) measurement of each power mode.

Table 3-1 Power Consumption

Power Mode	Power Consumption			
	Test Condition	Typical	Units	Resume Time
Shut Down (Single-Die)	CHIP_EN keeps Low	10	uA@3.3V	NA
Deep Sleep	1. LP Timer 2. WLAN OFF 3. Internal 0KB SRAM Data Retention	30	uA@3.3V	78.6ms
Standby	1. Real-M300 is power-gated 2. LP Timer 3. WLAN OFF 4. Internal 256KB SRAM Data Retention	200	uA@3.3V	17.5ms
Sleep	1. Real-M300 is clock-gated 2. LP Timer 3. WLAN OFF 4. Internal 256KB SRAM Data Retention	450	uA@3.3V	3.3ms
MCU Active	1. Real-M300 is in Active state 2. WLAN Off 3. Internal 256KB SRAM is retained	9	mA@3.3V	NA
Snooze Mode (for receiving WLAN BCN)	1. Real-M300 is clock-gated/power-gated 2. WLAN ON	58	mA@3.3V	NA

Measurement conditions: RF (SWR mode), VA11_SYN pin is NC, wake up by 32K timer, room temperature

3.2 Memory Control

Memory power control mechanism is also included in Ameba-Z II to control SRAM. Then user can set SRAM Block (The size of Real-M300 RAM is 256K Bytes) to low power status for power saving. Memory control only has two modes: normal mode and deepsleep mode (DS). For deepsleep mode, the RAM access pin value is pulled low when RAM enters this mode. After exiting this mode, content of RAM is unchanged.

3.3 Hardware Status of Power Modes

Table 3-2 shows all power mode status, the user can see each power mode correspond to each function status. The introduction of each power mode wake flow and wake source will be in the following sections.

In the table, **ON** status means that the corresponding hardware function could be powered and application software could make each hardware be disabled (OFF) if it would not be used in application strategy.

Table 3-2 Hardware Status of Power Mode

Functions	Active	Snooze	Sleep	Standby	Deep sleep	Shut Down
Real-M300 Core	ON	Clock-gated or Power-gated	Clock-gated	Power-gated	Power-gated	Power-gated
System Clock	ON	Clock-gated or Power-gated	Clock-gated	Power-gated	Power-gated	Power-gated
Internal SRAM	ON	ON/Deep-Sleep	ON	Deep Sleep	OFF	OFF
SWR_CORE	ON	ON	ON	ON	OFF	OFF
LDO	ON	ON	ON	ON	ON	OFF
LDO_BAT	ON	ON	ON	ON	ON	ON
IP_SEC	ON	OFF	OFF	OFF	OFF	OFF
I2C	ON	OFF	OFF	OFF	OFF	OFF
SPI	ON	OFF	OFF	OFF	OFF	OFF
DMAC	ON	OFF	OFF	OFF	OFF	OFF
UART1~2	ON	OFF	OFF	OFF	OFF	OFF
SDIO 2.0 Device	ON	ON	ON	OFF	OFF	OFF
UART0	ON	ON	ON	ON	OFF	OFF
WLAN	ON	ON	ON/Power saving mode	ON/Power saving mode	OFF	OFF
PWM	ON	ON	ON	ON	OFF	OFF
HS Timer	ON	ON	ON	ON	OFF	OFF
Low Precision Timer	ON	ON	ON	ON	ON	OFF
Wake Pin	ON	ON	ON	ON	ON	OFF

Note(*) pSRAM

3.4 Shutdown Mode

- CHIP_EN de-asserts to shutdown whole chip without external power cut components required. Once CHIP_EN is high the system will force into active mode.
- The shutdown mode status can refer to Table 3-2.

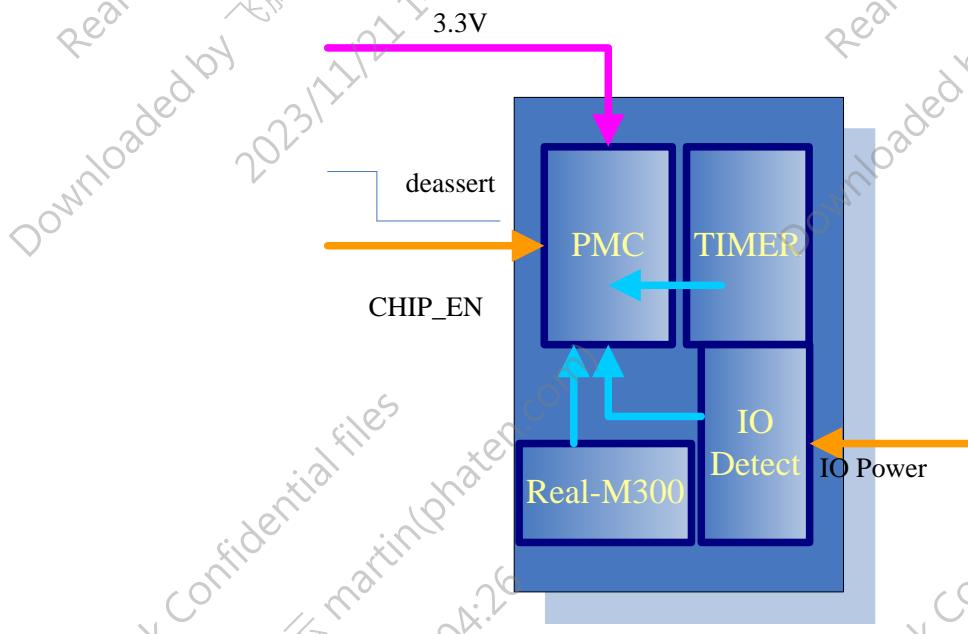


Figure 3-3 Shutdown Mode

3.5 Deep Sleep Mode

- CHIP_EN keeps high. User can invoke Deep Sleep API to force into deep sleep mode. By using specified interrupts (See Table 3-3) to wake up system.
- The following wake flow: Wake up ISR is high -> PMC -> enable CPU -> Reboot flow.
- The deep-sleep mode status can refer to Table 3-2.

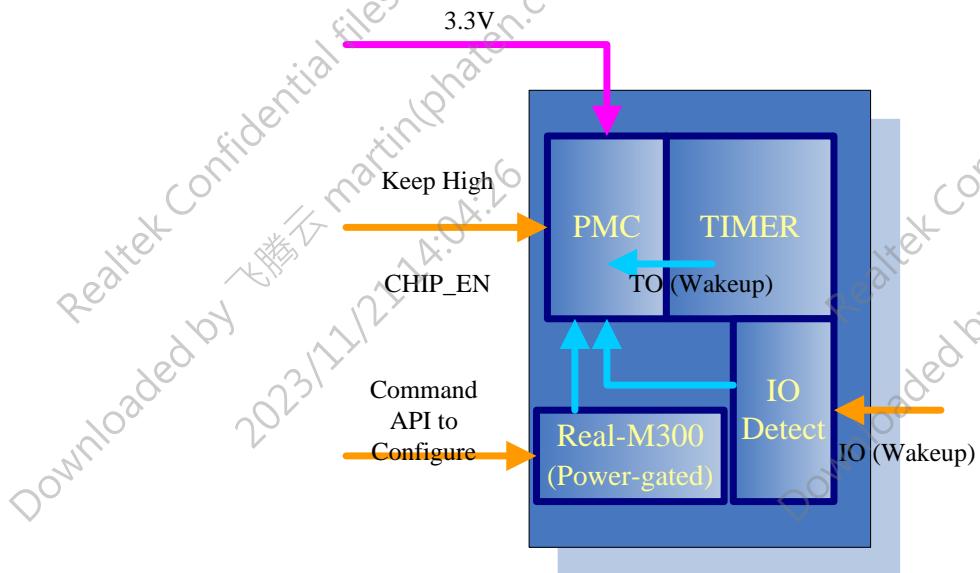


Figure 3-4 Deep Sleep Mode

3.5.1 Wakeup Source

Table 3-3 Deep Sleep Mode Wakeup Source

Wakeup Source	Note
Low Precision Timer	
Wake Pin	GPIOA_0 GPIOA_1 GPIOA_2 GPIOA_3 GPIOA_4 GPIOA_7 (depend on package) GPIOA_8 (depend on package) GPIOA_9 (depend on package) GPIOA_10 (depend on package) GPIOA_11 (depend on package) GPIOA_12 (depend on package) GPIOA_13 GPIOA_14 GPIOA_15 GPIOA_16 GPIOA_17 GPIOA_18 GPIOA_19 GPIOA_20 GPIOA_23

3.6 Standby Mode

- CHIP_EN keeps high. User can invoke Standby API to force into deep sleep mode. By using specified interrupts (See Table 3-4) to wake up system.
- The following wake flow: Wake up ISR is high -> PMC -> enable CPU -> Fast reboot flow.
- The standby mode status can refer to Table 3-2.

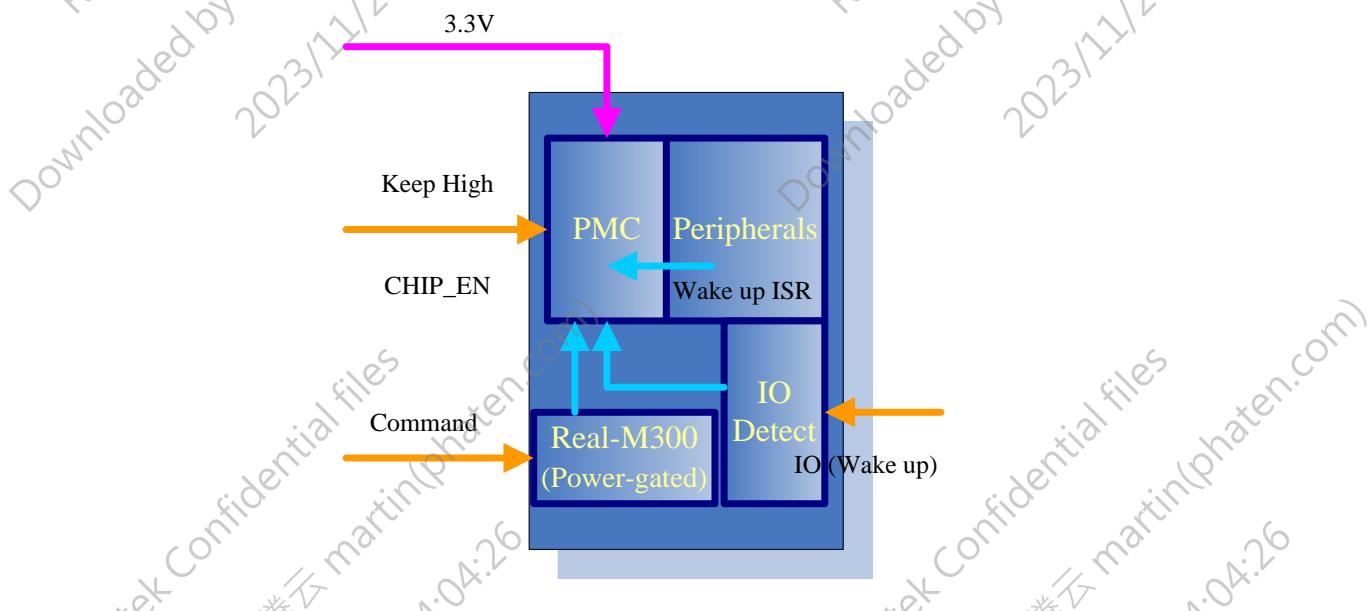


Figure 3-5 Standby Mode

3.6.1 Wakeup Source

Table 3-4 Standby Mode Wakeup Source

Wakeup Source	Note
Low Precision Timer	
Wake Pin	GPIOA_0 GPIOA_1 GPIOA_2 GPIOA_3 GPIOA_4 GPIOA_7 (depend on package) GPIOA_8 (depend on package)

	GPIOA_9 (depend on package)
	GPIOA_10 (depend on package)
	GPIOA_11 (depend on package)
	GPIOA_12 (depend on package)
	GPIOA_13
	GPIOA_14
	GPIOA_15
	GPIOA_16
	GPIOA_17
	GPIOA_18
	GPIOA_19
	GPIOA_20
	GPIOA_23
UART0	
WLAN	
PWM	
HS Timer	

3.7 Sleep Mode

- CHIP_EN keeps high. User can invoke Sleep API to force into deep sleep mode. By using specified interrupts (See Table 3-5) to wake up system.
- The following wake flow: Wake up ISR is high -> PMC -> enable CPU -> Execution of instructions continues.
- The sleep mode status can refer to Table 3-2.

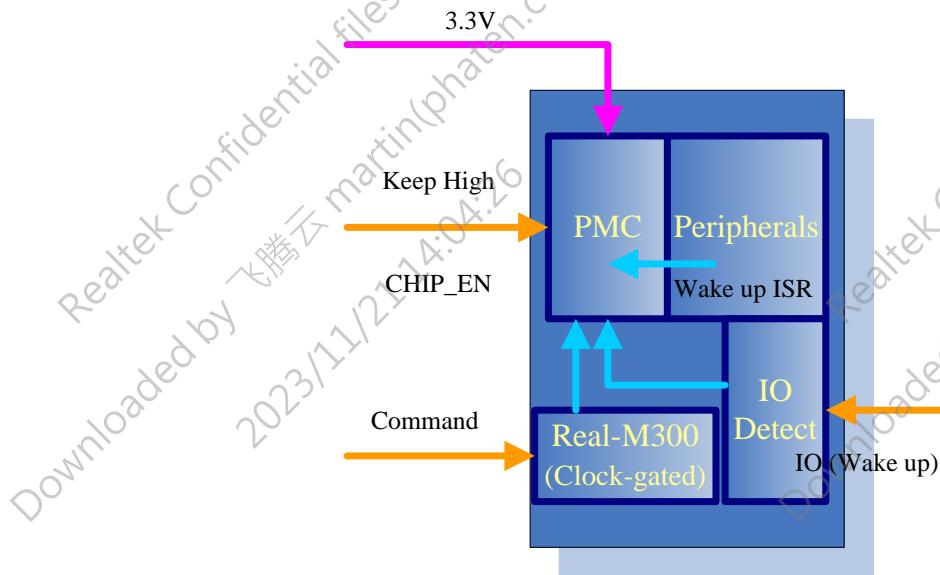


Figure 3-6 Sleep Mode

3.7.1 Wakeup Source

Table 3-5 Sleep Mode Wakeup Source

Wakeup Source	Note
Low Precision Timer	
Wake Pin	GPIOA_0 GPIOA_1 GPIOA_2 GPIOA_3 GPIOA_4 GPIOA_7 (depend on package) GPIOA_8 (depend on package) GPIOA_9 (depend on package) GPIOA_10 (depend on package) GPIOA_11 (depend on package) GPIOA_12 (depend on package) GPIOA_13

	GPIOA_14
	GPIOA_15
	GPIOA_16
	GPIOA_17
	GPIOA_18
	GPIOA_19
	GPIOA_20
	GPIOA_23
UART0	
WLAN	
PWM	
HS timer	
SDIO 2.0 Device	

3.8 Snooze Mode

- CHIP_EN keeps high. By using specified interrupts to wake up system.
- The following wake flow: WLAN power on request-> Receive particular beacon-> Wake up ISR is high -> PMC -> enable CPU -> Execution of instructions continues or fast reboot flow.
- The snooze mode status can refer to Table 3-2.

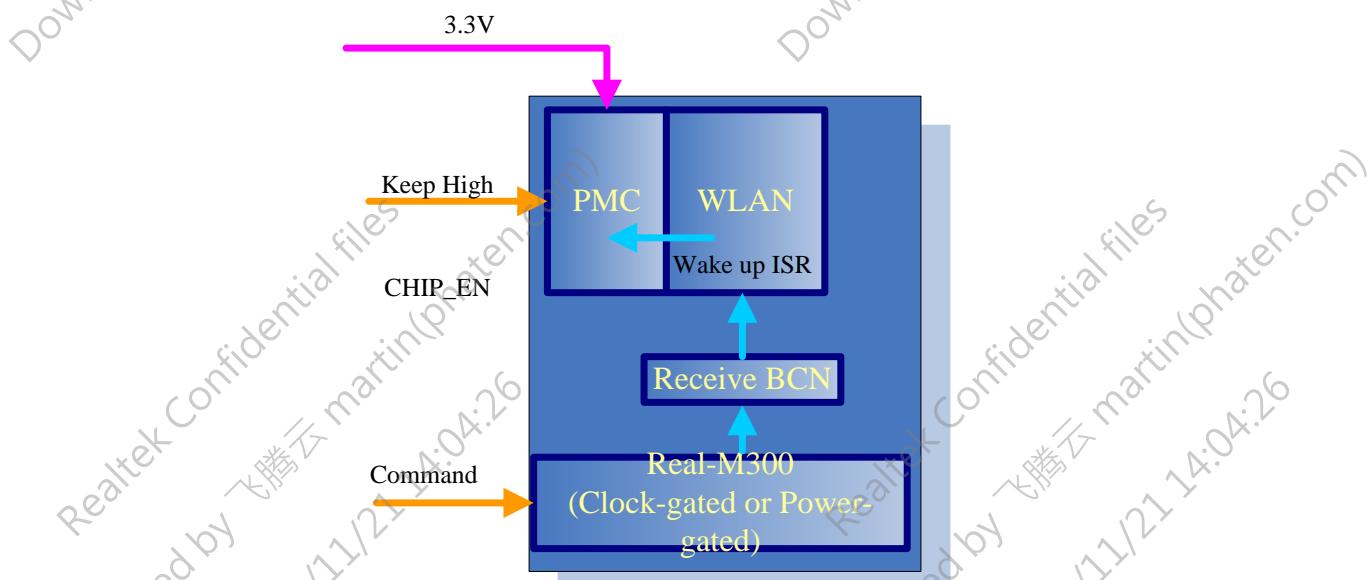


Figure 3-7 Snooze Mode

3.8.1 Wakeup Source

In snooze mode, the only wake up source is WLAN. The wakeup condition could be configured by WLAN driver according to system application. Once the event takes place, WLAN hardware would raise interrupt to PMC that could change hardware state.

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

4 Security Mode and Control

4.1 System Booting

Ameba-Z II system booting is divided as 2 or 3 stages:

- The 1st stage: On a platform reset event, for both cold reset and warm reset, the Ameba-Z II CPU will start the execution from the reset vector. And then the reset vector will redirect the program to the Reset Handler. This Reset Handler is an immutable ROM code which is located in the ITCM ROM. The Reset Handler will do the system hardware initialization, including power and clock configuration. And then it will start a firmware image loading. Ameba-Z II supports the firmware image loading from flash memory, SDIO interface or UART port.
- The 2nd stage: When the boot ROM finishes the firmware image loading, the program will jump to execute this firmware. If the firmware image is loaded from SDIO interface or UART port, the loaded firmware should be a user's application firmware. There is no next stage of booting. If the firmware image is loaded from the flash memory, this loaded firmware normally is a boot loader for user's application firmware loading. This boot loader will load user's application firmware image from flash.
- The 3rd stage: When the boot loader finish the firmware image loading, the program will jump to the entry function of user's application and start the program execution.

4.2 Secure Boot

The secure boot design of Ameba-Z II is based on following considerations:

- A clear text version of code is easily to be stolen and it is possible to be used for code analysis to find any existing security vulnerabilities. A method to defend these threats is firmware images are encrypted.
- A firmware image should be verified to make sure it didn't be tampered before loading or running of it. This verification default is implemented by HMAC (hash-based message authentication code). The boot loader will calculates the HMAC of a loading firmware and compare it with the HMAC from the firmware image.

Base on those considerations listed above, the secure boot mechanism is designed to have following features:

- A firmware image needs to be encrypted.
- Only a certificated firmware image can be loaded.

- Both key for encryption and certification are generated from the calculation of ECDH key exchange. The same private key of Ameba-Z II with different public key from different software module providers can get different shared keys for image decryption or certification.

4.2.1 Firmware Image Encryption and Verification

In Ameba-Z II secure boot mechanism, the secret key for a firmware image encryption/decryption or verification is generated from the calculation of ECDH key exchange cryptographic algorithm. There are two asymmetrical key pairs will be generated at the beginning. One private-public pair belongs to the Ameba-Z II; the other one belongs to the firmware image building tool. The private key of Ameba-Z II is provisioned into on-chip eFuse. From the ECDH key exchange scheme to derive a shared key, this shared key can be used as the secret key for encryption/decryption. The way to get secret key for firmware image verification is same as above. The difference is that they use different private keys.

4.2.2 Chain of Security

Ameba-Z II will load firmware images during the whole booting. To ensure each loaded component hasn't been tampered, the security of the system booting is originally from the immutable boot ROM code with the verification of the firmware images to be loaded. The boot ROM code verify the boot loader, the boot loader verify the next stage's firmware. This flow is called chain of security which make all of the loaded firmware be verified and trusted. Figure 4-1 shows the chain of security of Ameba-Z II booting for the firmware image loading from flash memory. The brief description of this flow is listed below:

- **Flash partition table decryption and loading:** The flash partition table is used to describe the location of each firmware image. The boot ROM will verify this partition table first. Because a tampered partition table will make the boot ROM code try to load an untrusted firmware image, the verification procedure of the partition table is inevitable. For secure consideration, the partition table can also be encrypted. The steps of the partition table decryption and loading are:
 - Use the "Enc Priv Key" and the "Dec Pub Key" from the partition table to calculate the shared key for the decryption of the partition table.
 - Decrypt the partition table and load it into internal RAM.
- **Flash partition table verification:**
 - Use the "Hash Priv Key" and the "Hash Pub Key" from the partition table to calculate

the shared key for the HMAC calculation.

- Calculate the HMAC of the partition table clear text.
 - Compare the HMAC calculation result. If the HAMC result is correct then continue the booting, otherwise the booting is failed.
- **Boot loader decryption and loading:** The processing of boot loader image decryption and loading is very similar to the flash partition table loading. The different is the public key for ECDH key exchange calculation is from the firmware image of the boot loader. For security consideration, the boot loader should always be loaded into internal RAM or an external memory with encryption.
- **Boot loader verification:** The method and flow of this verification is very similar to the verification of the flash partition table. The different is the public key for key exchange calculation is from the firmware image of the boot loader.
- **Verify the signature of the application firmware image:** The boot ROM checks the firmware image's signature which is within application firmware image to know whether a firmware image exist or not at flash memory with offset which is described by the partition table. There is a space which stores six public keys within an application firmware image. These public keys can be selected to be one of ECDH public key, so that customer can use different public keys with the only one private key to establish different shared secret keys which are used for further firmware images decryption. The steps of this verification are:
- If the application firmware image is encrypted, then decrypt the signature and the header part of the firmware image. The key for this decryption is generated from the ECDH key exchange calculation with "Enc Priv Key" and the public key within this firmware image.
 - Calculate the hash of the firmware image clear text. Compare the hash calculation result with the signature.
- **Application firmware image verification:** The verification steps are:
- Calculate the hash of the whole firmware, includes signature, 6 public keys, image header and image data. Compare the result of the hash calculation with the hash at the tail of the firmware image.
- **Application firmware image decryption and loading:** The steps of application firmware image description and loading are:
- The key for the decryption comes from the calculation of ECDH key exchange. There is a field of the image header gives the index to get the public key from the public key list within the firmware image. The shared key for image decryption is from the ECDH key exchange calculation with "Enc Priv Key" and the picked public key.
 - Decrypt the firmware image and load the image data to internal RAM and/or external RAM.

- An application firmware image contains more sub-images. The flow of decryption for each sub-image is the same as described above. Each sub-image has its own image header, and public key selection. Since different sub-image may come from different developer.
- A sub-image could be a flash XIP (execution in place) image. For this type of image, boot loader will configure the SCE hardware to do the XIP remapping and on-the-fly run time decryption.

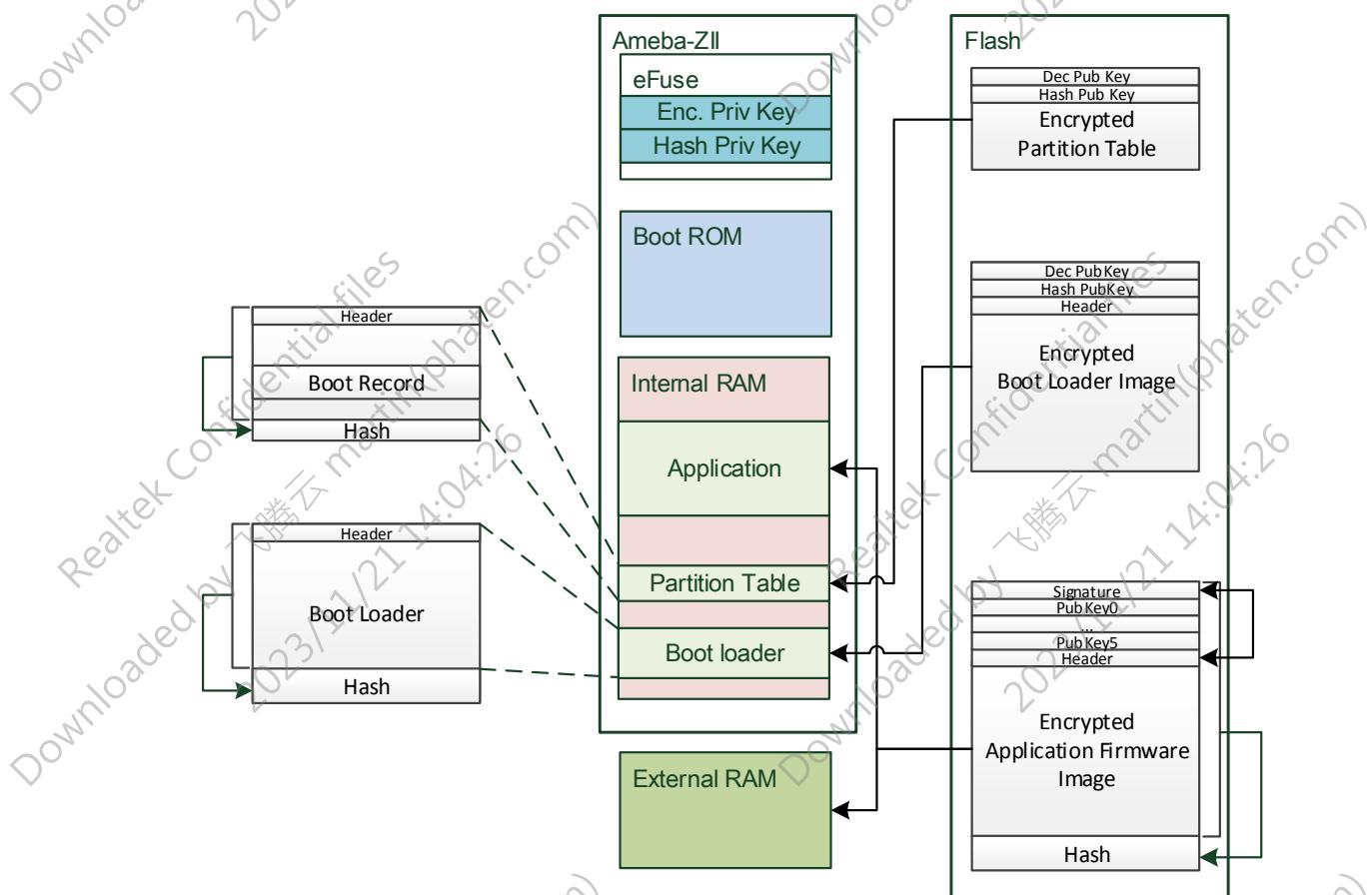


Figure 4-1 Ameba-Z II Secure Boot process with multiple stages

4.3 Ameba-Z II Secure features Overview

The Ameba-Z II provides below techniques to enhance system security:

- Hardware Crypto Accelerator
- Flash Memory Protection
- Debugger (JTAG/SWD) Protection

4.3.1 Functional Description

4.3.1.1 Hardware Crypto Accelerator

Ameba-Z II supports hardware crypto accelerator feature which is called Crypto Engine. During the chain of secure boot, each stage must verify the next. To reduce the boot time and prevent high value assets from being read by malicious software, with this feature, the firmware encryption/decryption and verification can be maintained with the correct security properties. Please reference Chapter 18 for more information of Crypto Engine.

4.3.1.2 Flash Memory Protection

Ameba-Z II supports flash memory execution in place (XIP) feature. With this feature, CPU can fetch instruction and data from flash memory directly and it does not need to load the firmware image into RAM executing this firmware. That means CPU needs to get a clear text from flash memory. However, a clear text version of firmware on flash could be harmful for the system security and also could be stolen easily. An IP, the Security Code Engine (SCE), is added to Ameba-Z II platform to solve this problem. SCE is a hardware IP which can do on-the-fly decryption for CPU to execute encrypted XIP code without performance compromised. If the XIP code is encrypted and the SCE decryption function is enabled; the SCE will read data from the flash and pass the decrypted data to CPU for code fetch of flash XIP.

The SCE control registers configuration normally is included in the system initialization. In Ameba-Z II SDK design, the boot loader will do this SCE configuration. For an encrypted XIP firmware image, the boot loader will decrypt the image header to get the key and initial vector (IV) for the SCE configuration. When an encrypted XIP firmware image be generated, the key and IV are decided. The firmware image builds tool use this key and IV to do the encryption of this XIP firmware image. The key and IV will be embedded into the image header for the boot loader can use the same key and IV to configure the SCE for XIP image decryption. Please reference Chapter 17 for more information of SCE.

4.3.1.3 Debugger (JTAG/SWD) Protection

The fundamental principles of debugging, which require access to the system state and system information, are in direct conflict with the principles of security, which require the restriction of access to assets. Ameba-ZII has a debugger port for platform software and hardware debugging. This debugger port is fully compliant to ARM CoreSight debugger architecture and support JTAG/SWD protocol. The fundamental principles of debugging, which require access to the system software and hardware, are in direct conflict with the principles of system security. So a restriction of the debugger function enable is required. Ameba-ZII implements this restriction by a debugger port protection mechanism (DPM). The protect mechanism will ensure that the JTAG is permitted access to those assets via a password method. Figure 4-2 shows the architecture of this DPM design.

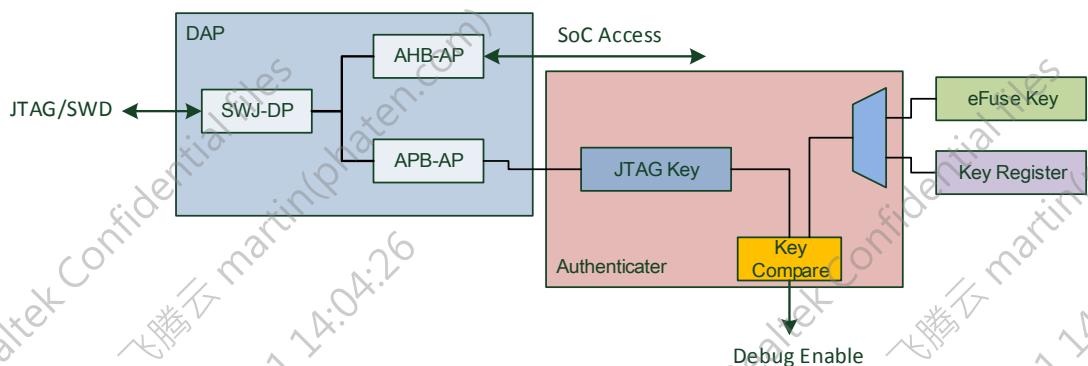


Figure 4-2 The Debugger Protection Mechanism Architecture.

When the debug port is locked (DPM authentication is enabled), it needs a procedure to enable the debug port function. The steps of this procedure are:

- Inject key from the debug port, the key data will be written to the DPM registers through APB-AP.
- Enable the key comparing. The correct key data can come from eFuse or from system registers optionally. If the key data is from the system registers, software is responsible to write correct key data to the system registers. The authenticator hardware will compare the key injected from the debug port with the correct key from eFuse or registers.
- The injected key pass the comparing, the AHB-AP will be enabled. The debugger can access the system software/hardware through this AHB-AP.

As this DPM design, the debugger function, with 3 states, is controlled by a sticky system register (can be written once only). This register can be written by software or hardware with value loading from eFuse. The states of debugger are listed as below:

State	Description
Enabled	The debug port is enabled. System debugging is allowed.
Disabled	The debug port is disabled. The debug port cannot be enabled.
Locked	The debug port is locked. It needs a un-lock procedure to make the state to transits from Locked state to Enabled state.

The debug port state transition is shown as Figure 4-3.

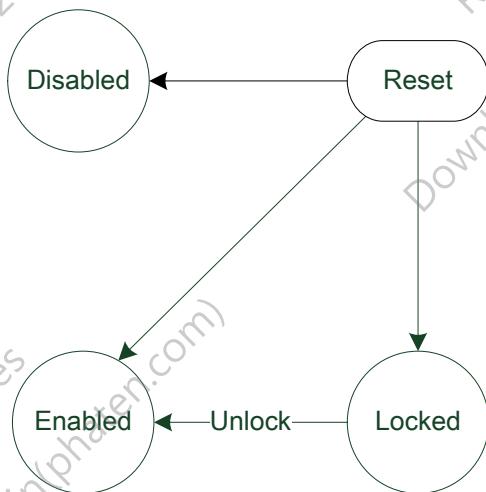


Figure 4-3 The Debugger State Transition.

4.4 Device Lifecycle State Management

Ameba-Z II boot ROM code implements a device lifecycle state to restrict part of software and hardware function for security consideration. In this implementation, the device lifecycle is divided as 4 stages. And a device lifecycle state is given to each stage. It is shown as Figure 4-4.

The device lifecycle state and its software or hardware restriction are listed below:

- Chip manufacture state: this is the default state of the device. All the debug and test function are available. There is no restriction for eFuse read/write. After a MP(mass production) test process and some of calibration data is programmed into the eFuse, the device lifecycle state is transited as the Device manufacture state.
- Device manufacture state: normally, OEMs do the product development and manufacture under this state. The debugger keep opened, the eFuse read/write still available in this state. After a device mass production process, the device lifecycle state is transited as the Deployed state. During the MP process, the “Enc Priv Key” and the “Hash Priv Key” should be programmed into the eFuse and the debugger port should either be disabled or enable

the JTAG function.

- Deployed state: The secure boot must be enabled in this state. Only verified firmware images can be loaded and executed. The debugger function is restricted, it either be disabled or need a unlock operation to enable it. The access to private key in eFuse also is restricted by eFuse lock configuration.
- RMA (Return Material Authorization) state: this state is used for the analysis of a failure device. The debugger is re-enabled to assist the analysis. The transition to this RMA state is through a token injection. The RMA state is the terminal state of the device lifecycle, no more state transition available.

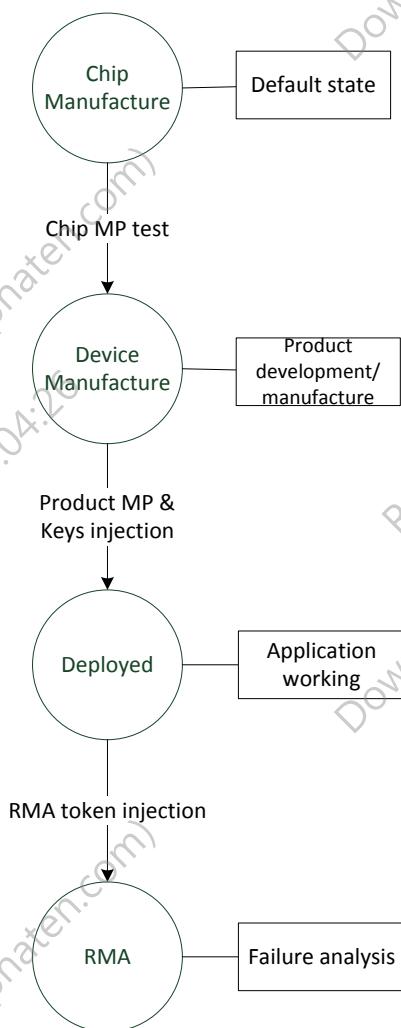


Figure 4-4 Ameba-Z II Device Lifecycle State

Following table summarize the hardware and software restriction for different device lifecycle state.

State	JTAG	Flash PG/ROM Shell Command
Chip Manufacturing	JTAG On	On
Device Manufacturing/Development	JTAG On	On
Deployed	JTAG Lock	Off
RMA	The JTAG function enable/lock/ disable is configured by eFuse	On

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

5 Pad Control and Pinmux

Ameba-Z II Pad and pinmux control includes various pad types, pull control, shutdown control, driving strength adjustment, Schmitt trigger, pinmux for internal communication interfaces.

The pad types which consists of 3.3V IO, 5V IO, SDIO could provide system design flexibility according to IO power requirement and external device connection.

Target application of Ameba-Z II aims on 3.3V system therefore 3.3V IO pad type is applied for most of Ameba-Z II pads. On the other hands, some particular requirement of 5V IO would be not rare, for example: 5V UART. To fulfill this part, 5V pad type is also provided in Ameba-Z II. And for SDIO device interface speed, high speed IO pads are added to this SoC.

5.1 Pad Control

5.1.1 Pad Types

According to the pad attribute, related information is listed in Table 5-1 in which the driving strength could be adjusted by control registers. To accomplish SDIO communication, the dedicated pins are provided for this high speed interface.

Table 5-1 Pad Types

Pad Type	Loading(pF)	Driving(mA)	Pull Resistance(ohm)	Pin Name
Normal	15	4/6/12/16	~75K @3.3V	GPIOA_0, GPIOA_1, GPIOA_2, GPIOA_3, GPIOA_4, GPIOA_5, GPIOA_6, GPIOA_7, GPIOA_8, GPIOA_9, GPIOA_10, GPIOA_11, GPIOA_12, GPIOA_21, GPIOA_22, GPIOA_23
5V	15	4/8 @3.3V 10/20 @5V	~70K @3.3V ~100K @5V	GPIOA_13, GPIOA_14
SDIO	15	4/8	~50K @3.3V	GPIOA_15, GPIOA_16, GPIOA_17, GPIOA_18, GPIOA_19, GPIOA_20

5.1.2 Pad Pull Control

In Ameba-Z II pad design, the related control register about pull up/pull down resistance would allow system designer to use these internal pull resistance for external device connection instead of using external resistance. In some scenario, high impedance of pad status would be helpful for current leakage condition. Our pad control also offers high impedance setting for the requirement. Table 5-2 contains the register setup of each pad status.

Table 5-2 Pad Pull Control

Pad Status \ Register Bit	PULL_CTRL[1]	PULL_CTRL[0]
High Impedance	0	0
Pull Up	1	0
Pull Down	0	1

5.1.3 Pad Schmitt Trigger

All I/O pads include a Schmitt Trigger that would be selectively disabled by setting the GPIOAXX_SMT_EN bit in the pad control register. The characteristic of Schmitt Trigger is Table 5-3. GPIOAx_SMT_EN in each pad control register is used to decide enable this circuit or not.

Table 5-3 Characteristics of Schmitt Trigger

Parameter	Min.	Typ.	Max.	Units
Schmitt-trigger High Level	1.377	1.683	1.908	V
Schmitt-trigger Low Level	0.729	0.957	1.116	V

5.1.4 Pad Driving Strength

The adjustable pad driving strength of Ameba-Z II would make selecting external device much easier because hardware designer could choose a more suitable device and use a proper driving strength to control it. Table 5-4 shows the control register for normal and SDIO pad driving control. Table 5-5 shows the control register for 5V pad driving control.

Table 5-4 Pad Driving Control of Normal and SDIO Pad

Driving Strength(mA)	GPIOAx_DRIVING[1]	GPIOAx_DRIVING[0]
4	0	0
6	0	1

12	1	0
16	1	1

Table 5-5 Pad Driving Control of 5V Pad

Driving Strength(mA)	GPIOAx_DRIVING[0]
10	0
20	1

5.1.5 Pad Shut Down

All of Ameba-Z II IO pad contain a shut down control circuit which could isolate the supply power of a pad. The shut down circuit would be helpful when considering power management in a power budget sensitive system. In our design, the shut down control is done by group style. Each pin in the same group would be shut down or enabled simultaneously while GPIOA_14, GPIOA_15, GPIOA_19 and GPIOA_20 have their own shut down control. All the shut down could be set by writing 1 to the GPIO_XX_SHDN bit.

Table 5-6 Pad Shut Down Control

Pad Shut Down Group	GPIO List
0	GPIOA_0, GPIOA_1, GPIOA_2, GPIOA_5, GPIOA_23
1	GPIOA_3, GPIOA_4, GPIOA_6, GPIOA_7, GPIOA_8, GPIOA_9, GPIOA_10, GPIOA_11, GPIOA_12
2	GPIOA_13, GPIOA_16, GPIOA_17, GPIOA_18, GPIOA_21, GPIOA_22

5.2 Pin Multiplexing Function

Ameba-Z II interface pins could be switched to a predefined pin sets by setting the GPIO control register. GPIOAx_PINMUX_SEL in each control register could choose which interface pin would occupy this GPIO pin. GPIOA0 pinmux selection is shown in Table 5-7 which indicates that GPIOA0 could be used for JTAG, UART1, EXT_32K, PWM0 and GPIO. All pins of Ameba-Z II have their own selection control register fields similar with Table 5-7.

Table 5-7 GPIOA0_PINMUX_SEL

GPIOA0_PINMUX_SEL[3:0]	Interface Selected
0000	RSVD

0001	JTAG_CLK
0010	UART1_IN
0011	EXT_32K
0100	RSVD
0101	PWM0
0110	RSVD
0111	GPIO

Overall pinmux table is given in Table 5-8. All of the pin would be used for GPIO interface.

Table 5-8 Pin Multiplexing Table

GPIO Pin Name	SPIC-Flash/ SDIO	JTAG	UART	SPI/WL_LED/ EXT_32K	I2C	PWM
GPIOA_0		JTAG_CLK	UART1_IN	EXT_32K		PWM[0]
GPIOA_1		JTAG_TMS	UART1_OUT	BT_LED		PWM[1]
GPIOA_2		JTAG_TDO	UART1_IN	SPI_CS _n	I2C_SCL	PWM[2]
GPIOA_3		JTAG_TDI	UART1_OUT	SPI_SCL	I2C_SDA	PWM[3]
GPIOA_4		JTAG_TRST	UART1_CTS	SPI_MOSI		PWM[4]
GPIOA_7	SPI_M_CS			SPI_CS _n		
GPIOA_8	SPI_M_CLK			SPI_SCL		
GPIOA_9	SPI_M_DATA[2]		UART0_RTS	SPI_MOSI		
GPIOA_10	SPI_M_DATA[1]		UART0_CTS	SPI_MISO		
GPIOA_11	SPI_M_DATA[0]		UART0_OUT		I2C_SCL	PWM[0]
GPIOA_12	SPI_M_DATA[3]		UART0_IN		I2C_SDA	PWM[1]
GPIOA_13			UART0_IN			PWM[7]
GPIOA_14	SDIO_INT		UART0_OUT			PWM[2]
GPIOA_15	SD_D[2]		UART2_IN	SPI_CS _n	I2C_SCL	PWM[3]
GPIOA_16	SD_D[3]		UART2_OUT	SPI_SCL	I2C_SDA	PWM[4]
GPIOA_17	SD_CMD					PWM[5]
GPIOA_18	SD_CLK					PWM[6]
GPIOA_19	SD_D[0]		UART2_CTS	SPI_MOSI	I2C_SCL	PWM[7]
GPIOA_20	SD_D[1]		UART2_RTS	SPI_MISO	I2C_SDA	PWM[0]
GPIOA_23				LED_0		PWM[7]

6 Real-M300 (KM4) CPU

The Real-M300 also called KM4 in former naming style is based on the ARMv8-M architecture that provides the advance system including single-cycle digital signal processing, low power consumption, etc. The Real-M300 adopts 3-stage instruction pipe and integrates “Instruction Cache Engine” (ICACHE) and “Data Cache Engine”(DCACHE) for the external memory including flash and pSRAM. It simplifies the design and software development of digital signal control systems with the integrated digital signal processing (DSP).

The Table 6-1 shows that Real-M 300 pays a little area to support many functions and performance.

Table 6-1 The Performance Table of Real-M300 vs CM33

CPU	Real-M300	CM33
Target Frequency(MHz)	100	50
Performance	DMIPS/MHz CoreMark/MHz	1.6 4.24
Support Feature	DSP FPU Icache Dcache	No Yes 32K 16K

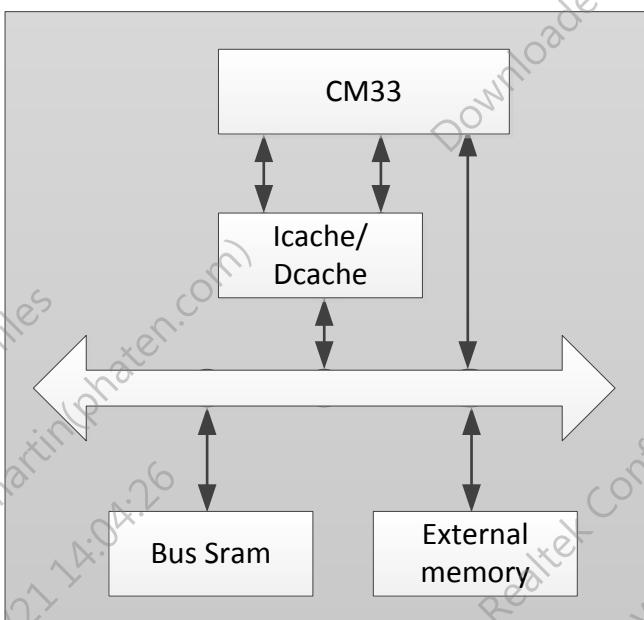


Figure 6-1 The System of CM33

From the view of the architecture, ICACHE and DCACHE engines are the main different between CM33 and Real-M300. The below figure could show the architecture separately. In the Figure 6-1, if the system with CM33 expects to get better performance for bus memory or the external memory, it needs integrate ICACHE and DCACHE. In the Figure 6-2, the system of real-M300 has included the ICACHE and DCACHE Engines and it can get better performance easily for system integrating view.

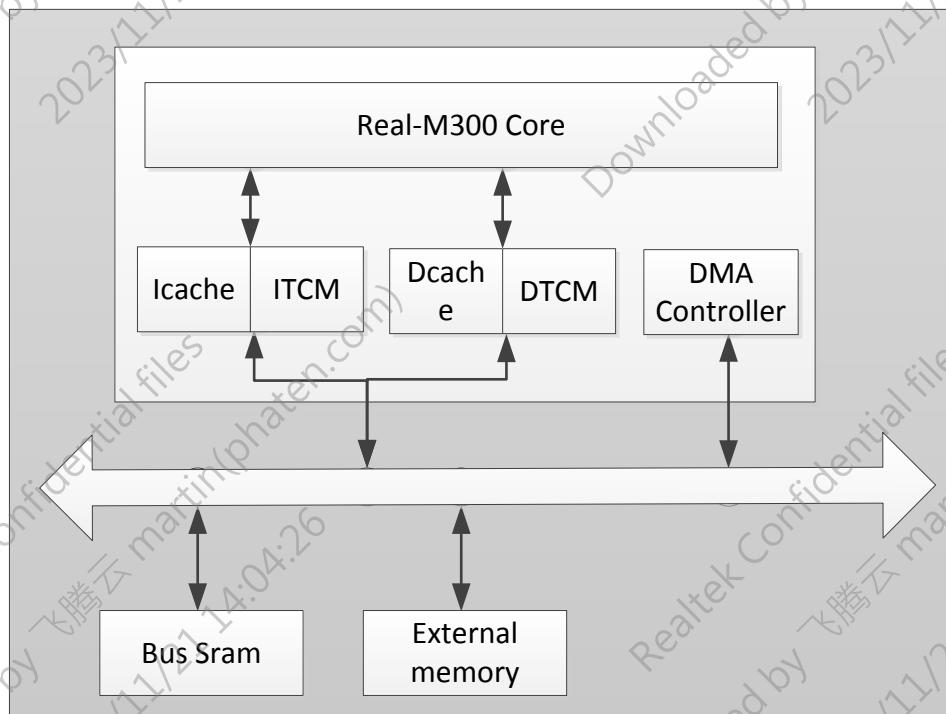


Figure 6-2 The System of Real-M300

6.1 Tightly Coupled Memory (TCM)

The architecture is Harvard, that is, storages and signal pathways are physically separated for instructions and data.

In parallel with each of the caches are two dedicated RAM storages accessible to both the instruction and data sides. These storages are Tightly Coupled Memories (TCMs).

Instruction TCM (ITCM) allows instructions to be fetched within one CPU cycle and Data TCM (DTCM) allows data to be read or written in one CPU cycle. There are up-to-2 TCMs can be configured both instruction and data sides. The regions for these TCMs are defined by the following registers:

Table 6-2 TCM Related Registers

Name	Address	Functions
ITCM0_BASE	0xE0042050	Defined the base address of ITCM0
ITCM0_TOP	0xE0042054	Defined the top address of ITCM0
ITCM1_BASE	0xE0042058	Defined the base address of ITCM1
ITCM1_TOP	0xE004205C	Defined the top address of ITCM1
DTCM0_BASE	0xE0042060	Defined the base address of DTCM0
DTCM0_TOP	0xE0042064	Defined the top address of DTCM0
DTCM1_BASE	0xE0042068	Defined the base address of DTCM1
DTCM1_TOP	0xE004206C	Defined the top address of DTCM1
ITCMCR	0xE000EF94	Enable control of ITCM0
ITCMCR1	0xE0042020	Enable control of ITCM1
DTCMCR	0xE000EF94	Enable control of DTCM0
DTCMCR1	0xE0042024	Enable control of DTCM1

Instruction fetches or data load store would check if the address is within the region defined by the registers in Table 6-3. If an instruction fetch address is in the ITCM region, it would fetch from ITCM directly. If not, it would continue to check if the instruction is in the Instruction Cache, otherwise fetch it from bus. If a data load store address is in the DTCM region, it would read from or write to the DTCM directly. If not, it would continue to check if the data is in the Data Cache, otherwise redirect the request to bus.

Although Real-M processor cache system is Harvard, Real-M300 has the same function; furthermore, it even allows instructions to be fetched from the DTCMs.

The following table shows the Real-M300 access priorities over ITCM, DTCM, Instruction Cache (ICache) and Data Cache (DCache):

Table 6-3 Real-M300 Data Read/Write Priority over TCMs and Caches

Instruction/Data	Hit ITCM (enabled)	Hit DTCM (enabled)	Hit ICache	Hit DCache	Behavior
Instruction fetch	Yes	-	-	-	Read from ITCM
	No	Yes*	-	-	Read from DTCM
	No	No	Yes	-	Read from ICache
	No	No	No	-	Read from bus
Data read/write	Yes	-	-	-	Read from/Write to ITCM
	No	Yes*	-	-	Read from/Write to DTCM
	No	No	-	Yes	Read from/Write to DCache
	No	No	-	No	Read from/Write to bus

Note(*) that, the function instruction fetch from DTCM is controlled/enabled by **I2DTCM** bit in **ACTLR (0xE000E008)** register while data load store from ITCM is controlled/enabled by **D2ITCM** bit in **ACTLR** register.

Although Real-M processors provides this access flexibility, instruction fetches from DTCM would still takes one more CPU cycle if the core is happened to executing a data load store instructions since it's Harvard architecture. The data read from ITCM would takes one more

CPU cycle as well if the core is happened to fetch an instruction.

Real-M processor provides TCM filling function to allow programmer to fill entire TCM with a store instruction. The control bits in **ACTLR** and their description are listed in Table 6-4:

Table 6-4 TCM Filling Control in ACTLR

Name	Bit	Description
ITCMFILLO	8	Fill entire ITCM0
ITCMFILL1	11	Fill entire ITCM1
DTCMFILLO	16	Fill entire DTCM0
DTCMFILL1	19	Fill entire DTCM1

6.2 CPU I/D Cache Engine

6.2.1 Introduction

The instruction cache and data cache are integrated into the input and output path of Real-M300. The performance of Real-M300 can get great improvement for the long latency of memories.

6.2.2 Function Description

The local memory system is configured during implementation and can include instruction and data caches of varying sizes. The cached instructions or data are fetched from external memory using the system bus interface. Any access that is not for a TCM is handled by the appropriate cache controller. If the access is to a Non-shareable cacheable memory, and the cache is enabled, a lookup is performed in the cache and, if found in the cache, that is, a cache hit, the data is fetched from or written into the cache. However, when the cache is not enabled, the access is Non-cacheable or is Shared memory then the accesses will directly read from or write to external through the system interface.

Both caches allocate a memory location to a cache line on a cache miss because of a read, that is, all cacheable locations are Read-Allocate in Real-M CPU. In addition, the data cache can allocate on a write access if the memory location is marked as Write-Allocate. When requesting a cache line, Real-M processor cache system supports desired-word-first policy and always fetches the requested data first, provides it to the core pipeline, and then fetches the rest of

the cache line. This enables the data read without waiting for the entire cache line to complete and is also known as critical word first.

The cache line size of Real-M CPU is 32-byte. That is, an instruction/data cache miss would request 32-byte from bus if the address is cacheable. The following table shows Real-M processor's read/write commands and corresponding results:

Table 6-5 Real-M Cache Behaviors

Command	Cacheability	Hit/Miss	Cache Policy	Results
Read	Cacheable	Hit	-	Return data to core-pipe
		Miss	-	Write-back when line dirty Read line from bus Return data to core-pipe
	UnCacheable	Hit	-	Invalidate when line clean Write-back invalidate when line dirty Read from bus Return data to core-pipe
		Miss	-	Read from bus Return data to core-pipe
Write	Cacheable	Hit	WT	Write to cache Write to bus
			WBA	Write to cache Mark as dirty if line is clean
		Miss	WT	Write to bus
			WBA	Read line from bus Write to cache Mark as dirty
	UnCacheable	Hit	-	Invalidate when line clean Write-back invalidate when line dirty Write to bus
		Miss	-	Write to bus

Note*: WT => Write-through

Note**: WBA => Write-back write-allocate

6.3 CPU System Tick Timer

6.3.1 Overview

6.3.1.1 Introduction

The system tick timer (SysTick) is integrated into the Real-M300 and the SysTick clock is fixed to half the frequency of the system clock.

6.3.1.2 Features

The features of system tick timer are showed in the below.

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

6.3.2 Functional Description

The SysTick timer is an integral part of the Real-M300. 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 SDM (32.768 kHz) clock. In order to generate recurring interrupts at a specific interval, the related register must be initialized with the correct value for the desired interval.

6.3.2.1 Overview

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 1000 Hz) 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.

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

6.4 Nested Vectored Interrupt Controller (NVIC)

6.4.1 Overview

The Real-M300 integrates a Nested Vectored Interrupt Controller (NVIC) that handling the settings of exceptions and interrupts. It achieves low-latency interrupt processing by the

flexibility settings that includes the priority and mask of the interrupts. Nested interrupt is also allowed.

6.4.1.1 Features

The features of NVIC is showed in the below.

- Supports 16 external interrupts
- Supports level-sensitive trigger type
- Supports 8 priority level
- Each interrupt can be enabled/disabled by writing to their corresponding register bit field

6.4.1.2 NVIC Table

Besides the default interrupts including IRQ number -15 to 0, the interrupts of other IPs are listed in the Table 6-6. In this table, the smallest IRQ number has the highest priority.

Table 6-6 NVIC Table

IRQ Number	Name	Description
-15	Reset	Reset vector, invoked on power up and warm reset
-14	NonMaskableInt	Non maskable interrupt, can't be stopped or preempted
-13	HardFault	Hard fault, all class of Fault
-12	MemoryManagement	Memory management, MPU mismatch, including Access Violation an No Match
-11	BusFault	Bus Fault, pre-fetch, memory access fault, other address/memory related fault
-10	UsageFault	Usage fault
-5	SVCall	System Service Call via SVC instruction
-4	DebugMonitor	Debug Monitor
-2	PendSV	Pendable request for system service
-1	SysTick	System Tick Timer
0	System	System
1	TimerGroup0	Timer Group0
2	TimerGroup1	Timer Group1
3	GPIO	GPIO
4	PWM	PWM
5	UART	UART
6	SPI	SPI
7	I2C	I2C
8	DMAC	DMAC
9	LowPower	LowPower
10	Crypto	Crypto
11	SDIO Device	SDIO Device
12	WLAN	WLAN

6.4.2 NVIC Control Registers

The related control registers for NVIC are as follows. For more detailed description, please refer to ARM's website.

Table 6-7 NVIC Related Control Registers

Name	Address	Description
NVIC_ISER<n>	0xE000E100	Interrupt Set Enable Register n
NVIC_ICER<n>	0xE000E180	Interrupt Clear Enable Register n
NVIC_ISPR<n>	0xE000E200	Interrupt Set Pending Register n
NVIC_ICPR<n>	0xE000E280	Interrupt Clear Pending Register n
NVIC_IABR<n>	0xE000E300	Interrupt Active Bit Register n
NVIC_IPR<n>	0xE000E400	Interrupt Priority Register n

6.5 Memory Protection Unit (MPU)

6.5.1 Overview

The Real-M300 processor includes the Memory Protection Unit (MPU) that provides full support for protection regions, access permissions and memory attributes. The MPU allows the privileged software to define memory regions with some memory attributes. Each region can't be overlapped and the region size must be aligned to multiple of 32 bytes. For the different memory attributes, tasks can be classified into the diffident behavior including disallowed accesses and read-only. If someone wants to execute the invalid action, MPU will trigger the exception. Real-M300 supports MPU entries.

Available memory attributes are as follows:

- The size of this region
- Is program execution allowed in this region?
- Access permissions (R/W)
- Cacheability
- Shareability for Normal memory

6.5.2 MPU Control Registers

The related control registers for MPU are as follows. For more detailed description, please refer to ARM's website.

Table 6-8 MPU Related Control Registers

Name	Address	Description
MPU_TYPE	0xE000ED90	MPU Type Register
MPU_CTRL	0xE000ED94	MPU Control Register
MPU_RNR	0xE000ED98	MPU Region Number Register
MPU_RBAR	0xE000ED9C	MPU Region Base Address Register
MPU_RLAR	0xE000EDA0	MPU Region Limit Address Register
MPU_RBAR_A< n >	0xE000EDA4	MPU Region Base Address Register Alias <i>n</i>
MPU_RLAR_A< n >	0xE000EDA8	MPU Region Limit Address Register Alias <i>n</i>
MPU_MAIRO	0xE000EDC0	MPU Memory Attribute Indirection Register 0
MPU_MAIR1	0xE000EDC4	MPU Memory Attribute Indirection Register 1

An optional **Memory Protection Unit (MPU)** allows programmer to define kinds of attributes. Through setting SCS registers **MAIRO** and **MAIR1**, up to 8 different lots of attribute could be defined. Here we named the attribute defined in **MAIR** as **attrfield** which is of 8-bit width. The following Table 6-9 is the recommended setting of **attrfield** in **MAIRs** for different types of memory.

Table 6-9 Recommended MPU Attrfield setting in MAIRO and MAIR1

Memory Type	Attrfield
Device Non-Bufferable	0x00 ~ 0x0B
Device Bufferable	0x0C ~ 0x0F
Non-Cacheable	0x44
Cacheable Write-Through	0x11, 0x22, 0x33, 0x88, 0x99, 0xAA, 0xBB
Cacheable Write-Back Write-Allocate	0x55, 0x77, 0xDD, 0xFF
Cacheable Write-Back Non-Write-Allocate	0x66, 0xCC, 0xEE

7 General Timer (GTIMER)

7.1 HS Timer

7.1.1 Overview

7.1.1.1 Application Scenario

The design of HS GTimer aims on low-power or battery-powered productions.

7.1.1.2 Features

- Timer mode
- Counter mode
 - Prescale
 - Load count
 - Current count
- Counting mode
 - Increment
 - Decrement
- Interrupt event
 - Time out
 - Match event
- Group Interrupt status

7.1.1.3 Architecture

A simplified block diagram of the component is illustrated in Figure 7-1 and Figure 7-2.

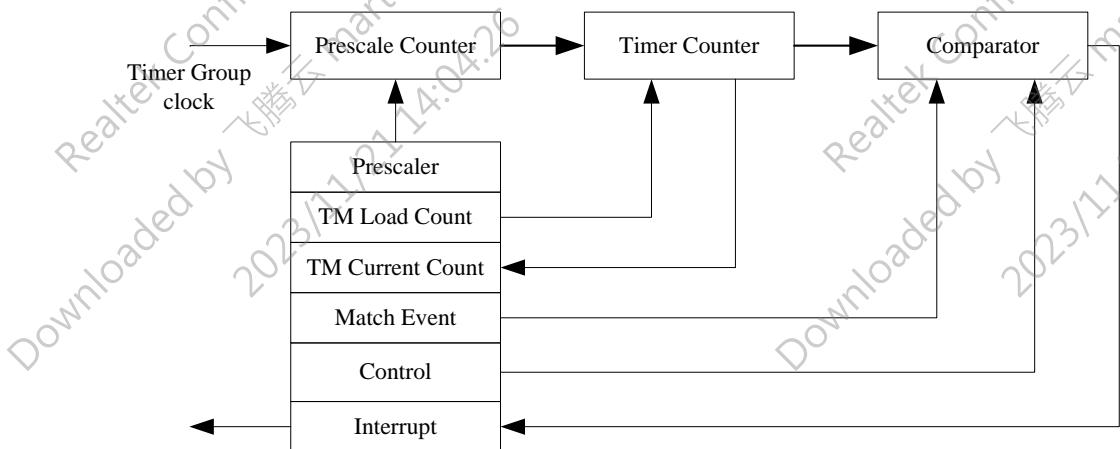


Figure 7-1 HS GTimer Function Block

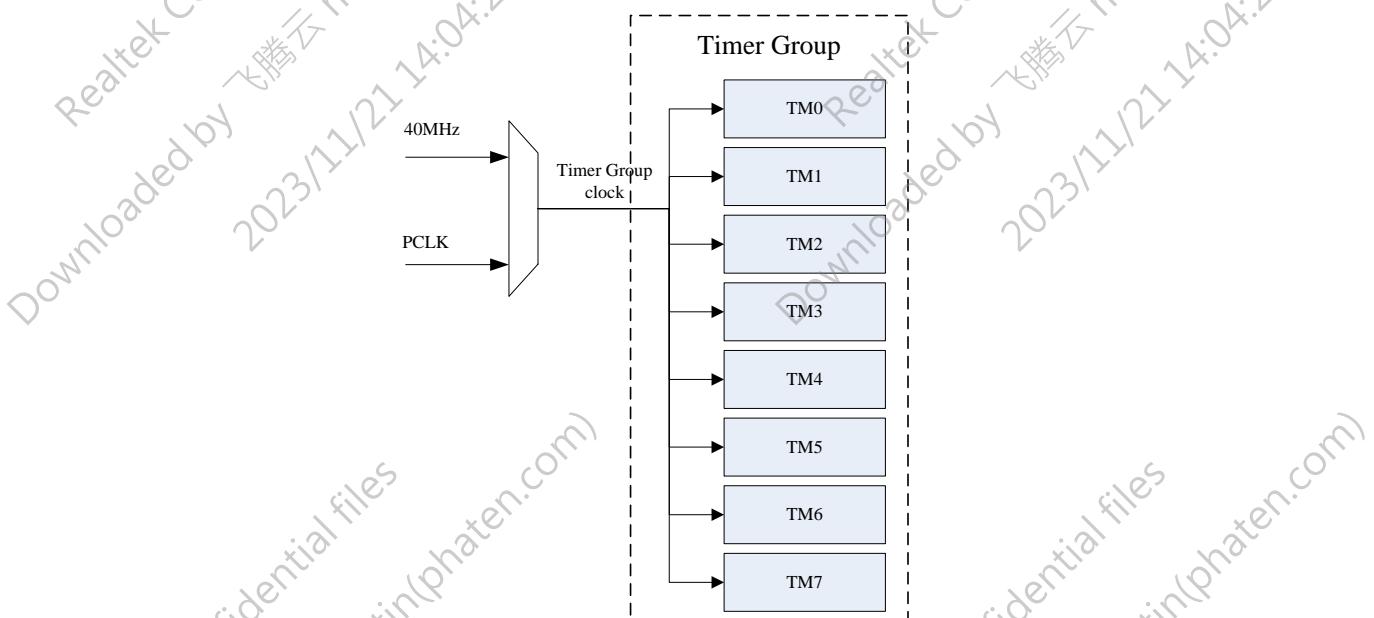


Figure 7-2 HS GTimer Architecture

7.1.2 Functional Description

HS GTimer only has 8 timer in the timer group and the source clock is from 40 MHz. In the Gtimer function, it provides Gtimer Mode Selection, counting mode, interrupts event and match event.

7.1.2.1 Mode Selection

The GTimer basically has two modes of operation are available:

■ Counting mode

- The timer continues to count after reaching default value, and generates an interrupt only one time. (To set “1” in the bit TMx_MOD of the Timerx control Register)

■ Periodic timer mode

- The counter generates an interrupt at a constant interval. (To set “0” in the bit TMx_MOD of the Timerx control Register)

7.1.2.2 Counting Mode

This timer is a 32-bit counter with its related auto-reload register. The counter can count up or count down. The counter clock can be divided by a 10-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 Gtimer counting mode includes:

■ Count up mode

- User can be set “0” in the bit TMx_CNT_MOD of the Timer0 control Register to enable up-counting mode for Gtimer.

■ Count down mode

- User can be set “1” in the bit TMx_CNT_MOD of the Timer0 control Register to enable down-counting mode for Gtimer.

Note: The TMx_LC Value to be loaded into TMx timer counter (TMx_TC) conditions are shown below:

- For down-counting, the “TMx_LC” Value to be loaded into TMx timer counters when TMx enable or timeout.
- For up-counting, the value at which timer timeout and loads 0 into TMx timer counter.

7.1.2.3 Match Event

In the Ameba-Z II HS GTimer also provide up to 4 match event for the user. User can use Timerx match event control Register (TMx_MECTRL) to select which match event want to use. The Timerx Match Event0 Counter Register (TMx_ME0~ TMx_ME3) is a 32-bit counter register.

7.1.2.4 Interrupt Event

In the Interrupts event provide time out event and match event, the following description was below:

- Time-out event
 - User can be set “1” in the bit TMx_IMR of the Timerx control Register to enable the time-out event.
- Match event
 - User can be set “1” in the Timerx match event control Register (TMx_MECTRL) to select which match event want to use.

The correspond interrupts event diagram was shown in Figure 7-3, which TM0_LC = 9.

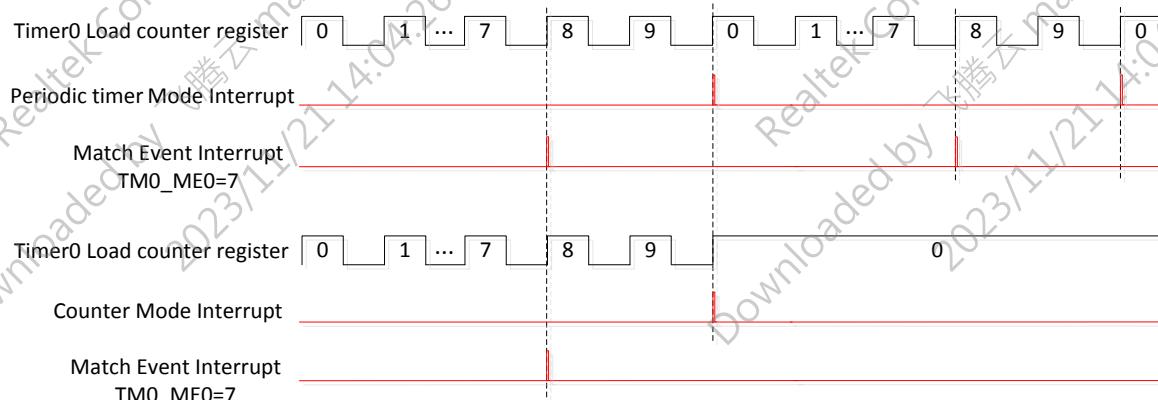


Figure 7-3 HS GTimer Interrupt Event Diagram

7.1.3 Control Registers

7.1.3.1 TG0_ISTS

- Name: Timer Group0 interrupt status Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-1 (Offset 0000h) TG0_ISTS

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	TG0_ISTS	0: the specific TM has no interrupt request 1: the specific TM has pending interrupt request

7.1.3.2 TG0_RAW_ISTS

- Name: Timer Group0 RAW interrupt status Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-2 (Offset 0004h) TG0_RAW_ISTS

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	TG0_RAW_ISTS	0: the specific TM has no interrupt request 1: the specific TM has pending interrupt request (permasking)

7.1.3.3 TG0_TMIR_CTRL

- Name: Timer Group0 indirect read control Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-3 (Offset 0008h) TG0_TMIR_CTRL

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7	W1S	0	POLL	Set this bit to enable indirect read current value of timer selected by TIMER_SEL. This bit is cleared by HW while finishing read and indicate REG_TIMER_TC is ready
6:5	R/W	0	rsvd	Reserve function
4	R/W	0	MODE	Mode selection. 0: Freq. of APB clock >> Freq. of Timer clock (about 10 times) 1: others

3	R/W	0	rsvd	Reserve function
2:0	R/W	0	TG0_TMIR_SEL	Timer group0 indirect read selection

7.1.3.4 TG0_TMIR_TC

- Name: Timer Group0 indirect read Timer Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-4 (Offset 000Ch) TG0_TMIR_TC

Bit	Access	INI	Symbol	Description
31:0	R	0	TG0_TMIR_TC	Current counter value of specific TM in REG_TG0_TMIR_CTRL

7.1.3.5 TM0_LC

- Name: Timer0 Load Count Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-5 (Offset 0040h) TM0_LC

Bit	Access	INI	Symbol	Description
31:0	R/W	0	TM0_LC	Value to be loaded into TM0 timer counter when TM0 enable or reset. Min = 0, when prescale register > 0; otherwise Min = 1.

7.1.3.6 TM0_TC

- Name: Timer0 Timer Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-6 (Offset 0044h) TM0_TC

Bit	Access	INI	Symbol	Description
31:0	R	0	TM0_TC	Current counter value of TM0.

7.1.3.7 TM0_PC

- Name: Timer0 Prescale Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-7 (Offset 0048h) TM0_PC

Bit	Access	INI	Symbol	Description
31:10	R	0	RSVD	
9:0	R	0	TM0_PC	Prescale Counter increases every clock after TM0 enabled. When the prescale counter is equal to PR, the next clock increments (decrements) the TC and clears the PC.

7.1.3.8 TM0_PR

- Name: Timer0 Prescale Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-8 (Offset 004Ch) TM0_PR

Bit	Access	INI	Symbol	Description
31:10	R	0	RSVD	
9:0	R/W	0	TM0_PR	When the PC is equal to this value, the next clock increments (decrements) the TC and clears the PC.

7.1.3.9 TM0_CTRL

- Name: Timer0 control Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-9 (Offset 0050h) TM0_CTRL

Bit	Access	INI	Symbol	Description
31:4	R	0	RSVD	
3	R/W	0	TMO_CNT_MOD	Counting mode for TMO 0: increment 1: decrement
2	R/W	0	TMO_IMR	Interrupt mask for TMO 0: disable interrupt 1: enable interrupt
1	R/W	0	TMO_MOD	Timer mode for TMO 0: timer mode; it reloads LC to TC and repeat the process according to the setting when the timeout event occurred. 1: counter mode; only one time process.

7.1.3.10 TM0_ISR

- Name: Timer0 interrupt status Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-10 (Offset 0054h) TM0_ISR

Bit	Access	INI	Symbol	Description
31:5	R	0	RSVD	
4	R/W1C	0	TMO_ME3_ISR	TMO match event3 interrupt
3	R/W1C	0	TMO_ME2_ISR	TMO match event2 interrupt
2	R/W1C	0	TMO_ME1_ISR	TMO match event1 interrupt
1	R/W1C	0	TMO_MEO_ISR	TMO match event0 interrupt
0	R/W1C	0	TMO_TO_ISR	TMO time out interrupt

7.1.3.11 TM0_MECTRL

- Name: Timer0 match event control Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-11 (Offset 005Ch) TM0_MECTRL

Bit	Access	INI	Symbol	Description
31:4	R	0	RSVD	
3	R/W	0	TMO_ME3_EN	TMO match event3 enable 0: Disable 1: Enable
2	R/W	0	TMO_ME2_EN	TMO match event2 enable 0: Disable 1: Enable
1	R/W	0	TMO_ME1_EN	TMO match event1 enable 0: Disable 1: Enable
0	R/W	0	TMO_MEO_EN	TMO match event0 enable 0: Disable 1: Enable

7.1.3.12 TM0_MEO

- Name: Timer0 Match Event0 Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-12 (Offset 0060h) TMO_ME0

Bit	Access	INI	Symbol	Description
31:0	R/W	0	TMO_ME0	Match event0 value of TMO.

7.1.3.13 TMO_ME1

- Name: Timer0 Match Event1 Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-13 (Offset 0064h) TMO_ME1

Bit	Access	INI	Symbol	Description
31:0	R/W	0	TMO_ME1	Match event1 value of TMO.

7.1.3.14 TMO_ME2

- Name: Timer0 Match Event2 Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-14 (Offset 0068h) TMO_ME2

Bit	Access	INI	Symbol	Description
31:0	R/W	0	TMO_ME2	Match event2 value of TMO.

7.1.3.15 TMO_ME3

- Name: Timer0 Match Event3Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-15 (Offset 006Ch) TMO_ME3

Bit	Access	INI	Symbol	Description
31:0	R/W	0	TMO_ME3	Match event3value of TMO.

7.1.3.16 TMx_Related

- Name: Timerx Related Register,x= 0 ~ 7
- Width: 32bit
- Initial Value: 0x0000

REG 7-16 (Offset (x+1)*0x40h) TMx_Related

Bit	Access	INI	Symbol	Description
31:0	R/W	0	RSVD	

7.2 LP Timer

7.2.1 Overview

7.2.1.1 Application Scenario

The design of LP GTimer aims on low-power or battery-powered productions.

7.2.1.2 Features

- Timer mode
- Counter mode
 - Prescale
 - Load count
 - Current count
- Counting mode
 - Increment
 - Decrement
- Interrupt event
 - Time out
 - Match event
- Group interrupt status

7.2.1.3 Architecture

A simplified block diagram of the component is illustrated in Figure 7-4 and Figure 7-5.

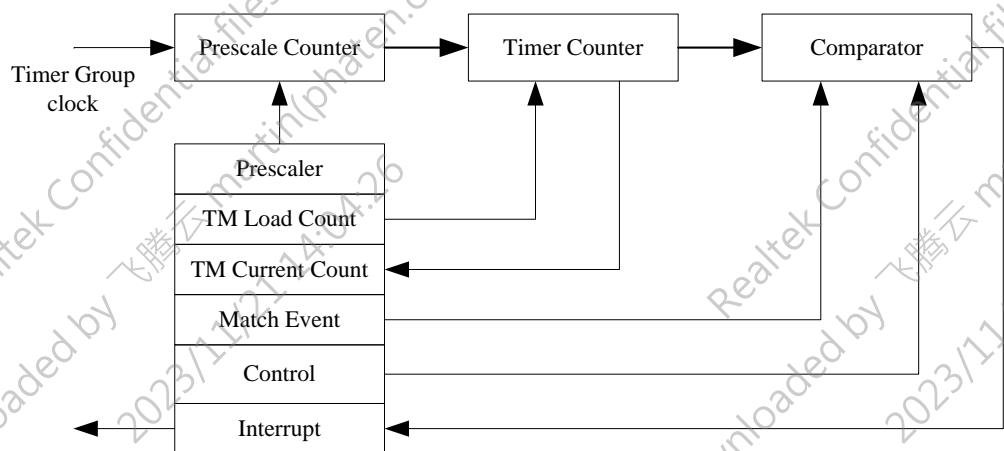


Figure 7-4 LP GTimer Function Block

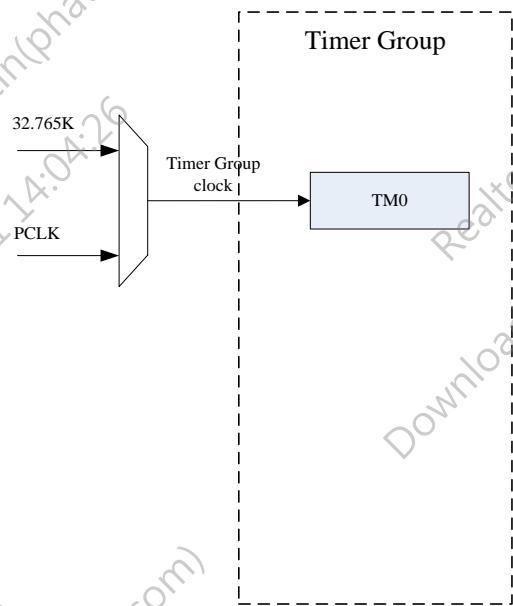


Figure 7-5 LP GTimer Architecture

7.2.2 Functional Description

LP GTimer only has TM0 in the timer group and the source clock is from 32 kHz. In the Gtimer function, it provides Gtimer Mode Selection, counting mode, interrupts event and match event.

7.2.2.1 Mode Selection

The GTimer basically has two modes of operation are available:

- Counter mode
 - The timer continues to count after reaching default value, and generates an interrupt only one time. (To set “1” in the bit TM0_MOD of the Timer0 control Register)
- Periodic timer mode
 - The counter generates an interrupt at a constant interval. (To set “0” in the bit TM0_MOD of the Timer0 control Register)

7.2.2.2 Counting Mode

This timer is a 32-bit counter with its related auto-reload register. The counter can count up or count down. The counter clock can be divided by a 10-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 Gtimer counting mode includes:

- Count up mode
 - User can be set “0” in the bit TM0_CNT_MOD of the Timer0 control Register to enable up-counting mode for Gtimer.
- Count down mode
 - User can be set “1” in the bit TM0_CNT_MOD of the Timer0 control Register to enable down-counting mode for Gtimer.

Note: The TM0_LC value to be loaded into TM0 timer counter (TM0_TC) conditions are shown below:

- For down-counting, the “TM0_LC” value to be loaded into TM0 timer counters when TM0 enable or timeout.
- For up-counting, the value at which timer timeout and loads 0 into TM0 timer counter.

7.2.2.3 Match Event

In the Ameba-Z II LP GTimer also provide up to 4 match event for the user. User can use Timer0 match event control Register (TM0_MECTRL) to select which match event want to use. The Timerx Match Event0 Counter Register (TM0_ME0~ TM0_ME3) is a 32-bit counter register.

7.2.2.4 Interrupt Event

In the Interrupts event provide time out event and match event, the following description was below:

- Time-out event
 - User can be set “1” in the bit TM0_IMR of the Timer0 control register to enable the time-out event.
- Match event
 - User can be set “1” in the Timer0 match event control Register (TM0_MECTRL) to select which match event want to use.

The correspond interrupts event diagram was shown in Figure 7-6.

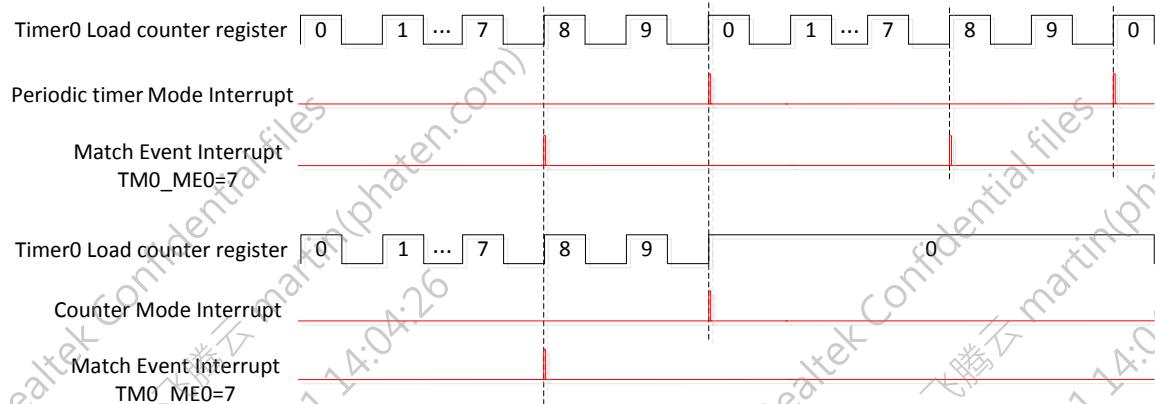


Figure 7-6 LP GTimer Interrupt Event Diagram

7.2.3 Control Registers

7.2.3.1 TG0_ISTS

- Name: Timer Group0 interrupt status Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-17 (Offset 0000h) TG0_ISTS

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	TG0_ISTS	0: the specific TM has no interrupt request 1: the specific TM has pending interrupt request

7.2.3.2 TG0_RAW_ISTS

- Name: Timer Group0 RAW interrupt status Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-18 (Offset 0004h) TG0_RAW_ISTS

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	TG0_RAWISTS	0: the specific TM has no interrupt request 1: the specific TM has pending interrupt request (permasking)

7.2.3.3 TG0_TMIR_CTRL

- Name: Timer Group0 indirect read control Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-19 (Offset 0008h) TG0_TMIR_CTRL

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7	W1S	0	POLL	Set this bit to enable indirect read current value of timer selected by TIMER_SEL. This bit is cleared by HW while finishing read and indicate REG_TIMER_TC is ready
6:5	R/W	0	rsvd	Reserve function
4	R/W	0	MODE	Mode selection. 0: Freq. of APB clock >> Freq. of Timer clock (about 10 times) 1: others
3	R/W	0	rsvd	Reserve function
2:0	R/W	0	TG0_TMIR_SEL	Timer group0 indirect read selection

7.2.3.4 TG0_TMIR_TC

- Name: Timer Group0 indirect read Timer Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-20 (Offset 000Ch) TG0_TMIR_TC

Bit	Access	INI	Symbol	Description
31:0	R	0	TG0_TMIR_TC	Current counter value of specific TM in REG_TG0_TMIR_CTRL

7.2.3.5 TM0_LC

- Name: Timer0 Load Count Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-21 (Offset 0040h) TM0_LC

Bit	Access	INI	Symbol	Description
31:0	R/W	0	TM0_LC	Value to be loaded into TM0 timer counter when TM0 enable or reset. Min = 0, when prescale register > 0; otherwise Min = 1.

7.2.3.6 TM0_TC

- Name: Timer0 Timer Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-22 (Offset 0044h) TM0_TC

Bit	Access	INI	Symbol	Description
31:0	R	0	TM0_TC	Current counter value of TM0.

7.2.3.7 TM0_PC

- Name: Timer0 Prescale Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-23 (Offset 0048h) TM0_PC

Bit	Access	INI	Symbol	Description
31:10	R	0	RSVD	
9:0	R	0	TM0_PC	Prescale Counter increases every clock after TM0 enabled. When the prescale counter is equal to PR, the next clock increments (decrements) the TC and clears the PC.

7.2.3.8 TM0_PR

- Name: Timer0 Prescale Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-24 (Offset 004Ch) TM0_PR

Bit	Access	INI	Symbol	Description
31:10	R	0	RSVD	
9:0	R/W	0	TMO_PR	When the PC is equal to this value, the next clock increments (decrements) the TC and clears the PC.

7.2.3.9 TM0_CTRL

- Name: Timer0 control Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-25 (Offset 0050h) TM0_CTRL

Bit	Access	INI	Symbol	Description
31:4	R	0	RSVD	
3	R/W	0	TMO_CNT_MOD	Counting mode for TM0 0: increment 1: decrement
2	R/W	0	TMO_IMR	Interrupt mask for TM0 0: disable interrupt 1: enable interrupt
1	R/W	0	TMO_MOD	Timer mode for TM0 0: timer mode; it reloads LC to TC and repeat the process according to the setting when the timeout event occurred. 1: counter mode; only one time process.

7.2.3.10 TM0_ISR

- Name: Timer0 interrupt status Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-26 (Offset 0054h) TMO_ISR

Bit	Access	INI	Symbol	Description
31:5	R	0	RSVD	
4	R/W1C	0	TMO_ME3_ISR	TMO match event3 interrupt
3	R/W1C	0	TMO_ME2_ISR	TMO match event2 interrupt
2	R/W1C	0	TMO_ME1_ISR	TMO match event1 interrupt
1	R/W1C	0	TMO_MEO_ISR	TMO match event0 interrupt
0	R/W1C	0	TMO_TO_ISR	TMO time out interrupt

7.2.3.11 TMO_MECTRL

- Name: Timer0 match event control Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-27 (Offset 005Ch) TMO_MECTRL

Bit	Access	INI	Symbol	Description
31:4	R	0	RSVD	
3	R/W	0	TMO_ME3_EN	TMO match event3 enable 0: Disable 1: Enable
2	R/W	0	TMO_ME2_EN	TMO match event2 enable 0: Disable 1: Enable
1	R/W	0	TMO_ME1_EN	TMO match event1 enable 0: Disable 1: Enable
0	R/W	0	TMO_MEO_EN	TMO match event0 enable 0: Disable 1: Enable

7.2.3.12 TMO_MEO

- Name: Timer0 Match Event0 Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-28 (Offset 0060h) TMO_MEO

Bit	Access	INI	Symbol	Description
31:0	R/W	0	TMO_MEO	Match event0 value of TMO.

7.2.3.13 TMO_ME1

- Name: Timer0 Match Event1 Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-29 (Offset 0064h) TMO_ME1

Bit	Access	INI	Symbol	Description
31:0	R/W	0	TMO_ME1	Match event1 value of TMO.

7.2.3.14 TMO_ME2

- Name: Timer0 Match Event2 Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-30 (Offset 0068h) TMO_ME2

Bit	Access	INI	Symbol	Description
31:0	R/W	0	TMO_ME2	Match event2 value of TMO.

7.2.3.15 TMO_ME3

- Name: Timer0 Match Event3Counter Register
- Width: 32bit
- Initial Value: 0x0000

REG 7-31 (Offset 006Ch) TMO_ME3

Bit	Access	INI	Symbol	Description
31:0	R/W	0	TMO_ME3	Match event3value of TMO.

8 Watchdog Timer (WDT)

8.1 Overview

8.1.1 Application Scenario

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.

8.1.2 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 32.768 kHz
 - Timeout formula:

$$\text{Timeout} = \frac{1}{32,768\text{kHz}} * \text{Count}$$
$$\frac{\text{DivFactor} + 1}{}$$

8.1.3 Architecture

A simplified block diagram of the component is illustrated in Figure 8-1.

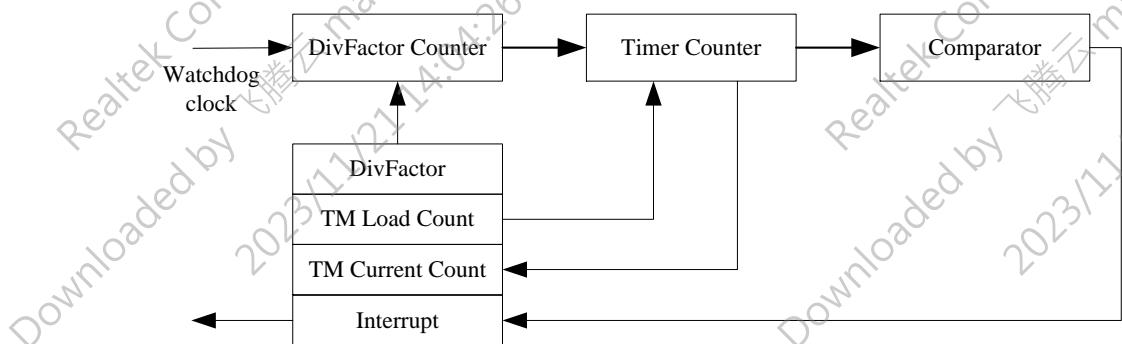


Figure 8-1 Watchdog Timer Architecture

8.2 Functional Description

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.

8.2.1 Operation Mode

The Watchdog timer basically has two modes of operation are available:

- Timeout mode
 - Reset system when WDT timer counter is overflow.
- Interrupt mode
 - Interrupt CPU when WDT timer counter is overflow.

8.3 Control Registers

8.3.1 WATCH_DOG_TIMER

- Name: Watch Dog Timer Control Register
- Width: 32bit
- Initial Value: 0x3010001

REG 8-1 (Offset 0000h) WATCH_DOG_TIMER

Bit	Access	INI	Symbol	Description
31	R/W1C	0	WDT_TO	Watch dog timer timeout. 1 cycle pulse
30	R/W	0	WDT_MODE	0: Interrupt CPU when WDT timer counter is overflow. 1: Reset system when WDT timer counter is overflow.
29	R	0	RSVD	
28:25	R/W	1	CNT_LIMIT	0: 0x001 1: 0x003 2: 0x007 3: 0x00F 4: 0x01F 5: 0x03F 6: 0x07F 7: 0xOFF 8: 0x1FF 9: 0x3FF 10: 0x7FF 11~15: 0xFFFF Watchdog Count= (0x00000001 << (CNT_LIMIT + 1)) - 1
24	W	1	WDT_CLEAR	Write 1 to clear timer
23:16	R/W	1	WDT_EN_BYTE	Set 0xA5 to enable watch dog timer
15:0	R/W	1	VNDR_DIVFACTOR	Dividing factor. Watch dog timer is count with 32.768KHz/(divfactor+1). Minimum dividing factor is 1.

9 Pulse Width Modulation (PWM)

9.1 Overview

9.1.1 Application Scenario

The design of PWM module aims on fading LED, server motor control.

9.1.2 Features

- PWM generator 8 sets with configurable duty ratio (0 ~ 100%)
- Each PWM have auto adjustment mode
 - Auto Increment
 - Auto Decrement
- Interrupt event
 - Period-end event
 - Auto adjustment event

9.1.3 Architecture

A simplified block diagram of the component is illustrated in Figure 9-1.

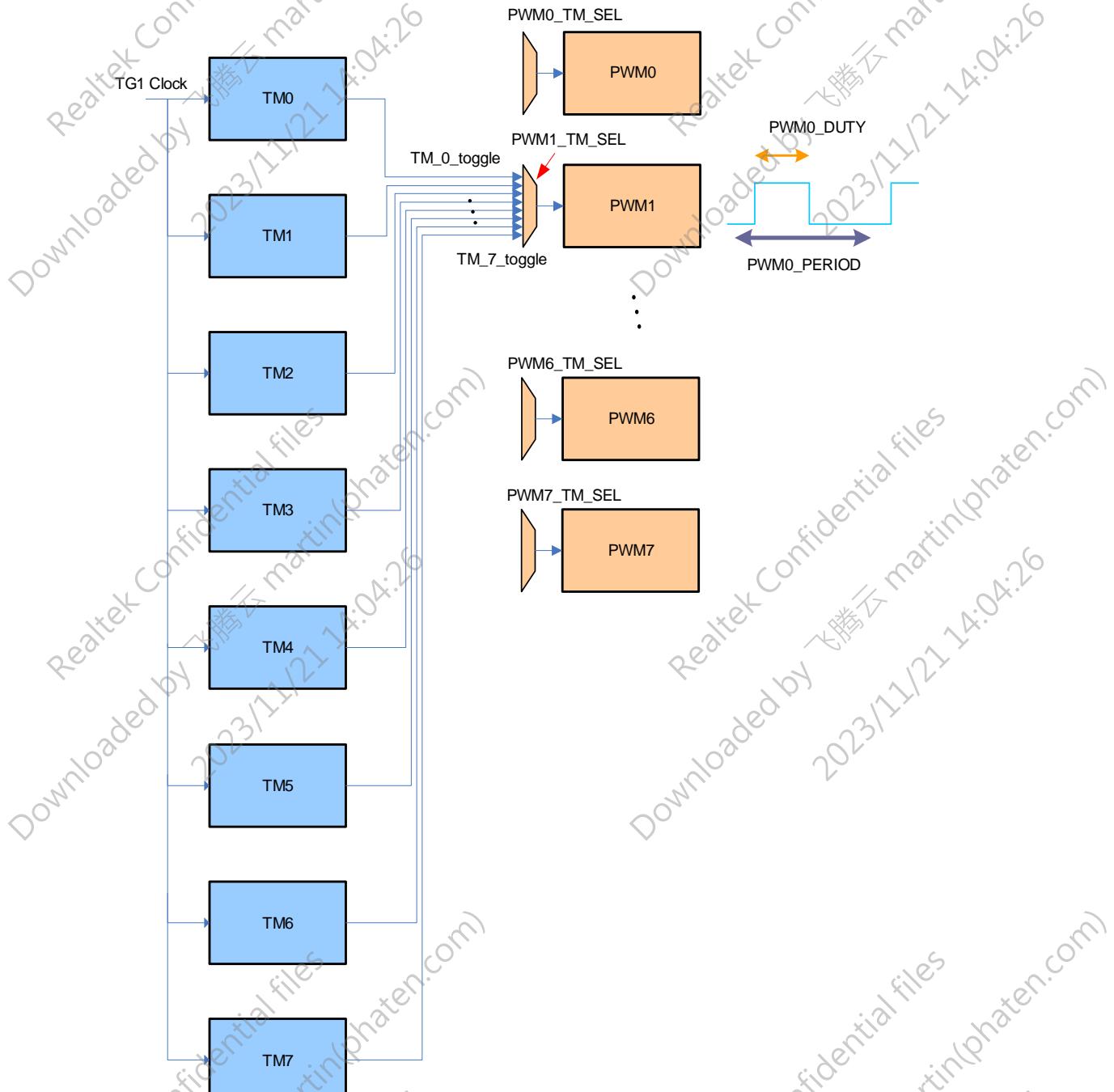


Figure 9-1 PWM Architecture

9.2 Functional Description

The PWM module usually for some applications, ex. fading LED, server motor control, the PWM duty needs to be adjusted by time fading. To offload the CPU loading, the PWM auto-adjustment hardware can auto adjust the duty ratio of the PWM without software interoperability.

9.2.1 PWM Mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the `PWMx_PERIOD` register and a duty cycle determined by the value of the `PWMx_DUTY` register.

- Period = $(\text{PWMx_PERIOD}+1) * T_{\text{CNT}}$
 - Duty = $(\text{PWMx_DUTY}+1) * T_{\text{CNT}}$
- Where $T_{\text{CNT}} = T_{\text{XTAL}} * (\text{PSC} + 1)$

The PWM mode can be selected independently on each channel by setting '1' in the `PWMx_CTRL_SET` bits in the `PWMx_CTRL` register. You must enable the corresponding GTimer register, and eventually the auto-reload preload register by setting in the `TMx_LC` register. The Figure 9-2 shows some PWM waveforms in an example where `PWMx_PERIOD` = 9.

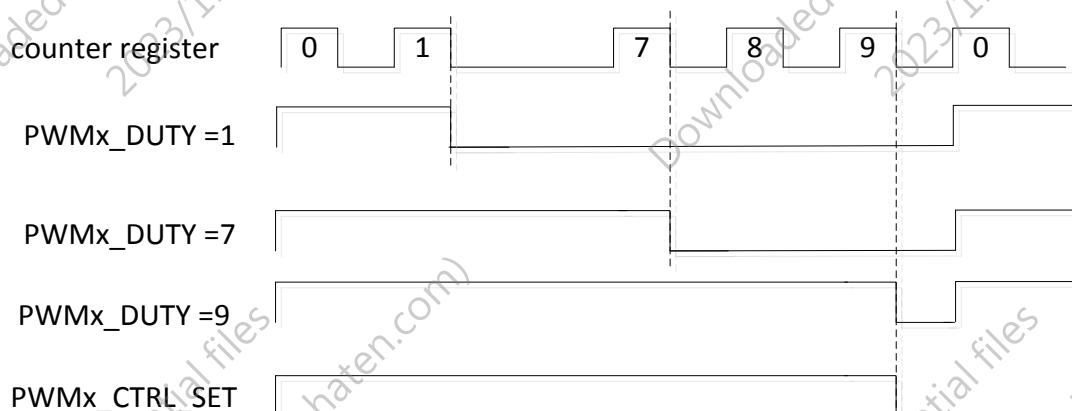


Figure 9-2 PWM Waveforms (`PWMx_PERIOD` = 9)

9.2.2 PWM Auto Adjustment

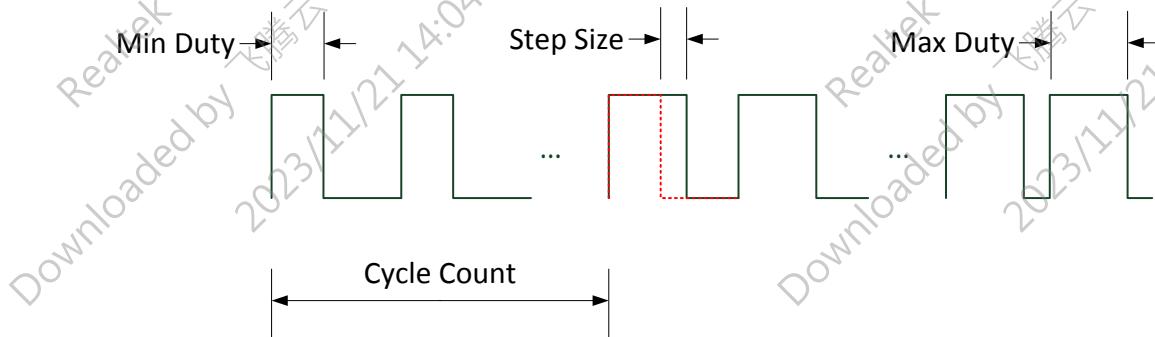


Figure 9-3 PWM Auto-Adjustment

The Figure 9-3 shows the behavior of the PWM auto-adjustment. There are few parameters need to be defined:

- Cycle Count: this parameter defines the duty ration adjustment speed. The duty ratio will be adjusted by increase a step of decrease a step every ‘Cycle Count’ of PWM period.
- Step Size: this parameter defines how much duty ratio will be increased or decreased by an adjustment. The Step size for duty ratio increasing may different with the step size for duty ratio decreasing.
- Min Duty: The down limit of the duty ratio. The HW will stop the duty ratio decreasing or reverse to start to increase the duty ratio when the duty reaches this value.
- Max Duty: The up limit of the duty ratio. The HW will stop the duty ratio increasing or reverse to start to decrease the duty ratio when the duty reaches this value.

9.2.3 PWM Interrupt Event

The Ameba-Z II provide the 2-type PWM interrupt event. The type of interrupt is programmable with one of the following settings:

- **Period-end event:** The “period-end event” will be enabled by setting “1” in the “PWMx_PERIOD_INT” register and trigger interrupt each cycle.
- **Auto adjustment event**

- **Auto adjustment up-limit event:** The “auto adjustment up-limit event” will be enabled by setting “1” in the DUTYx_ADJ_UP_INT register and trigger interrupt while achieving the up Limit of the Adjusted Duty (DUTYx_ADJ_UP_LIM).
- **Auto adjustment down-limit event:** The “auto adjustment down-limit event” will be enabled by setting “1” in the DUTYx_ADJ_DN_INT register and trigger interrupt while achieving the down Limit of the Adjusted Duty (DUTYx_ADJ_DWN_LIM).

Figure 9-4 shows PWM interrupt event diagram in an example where PWMx_PERIOD = 9.

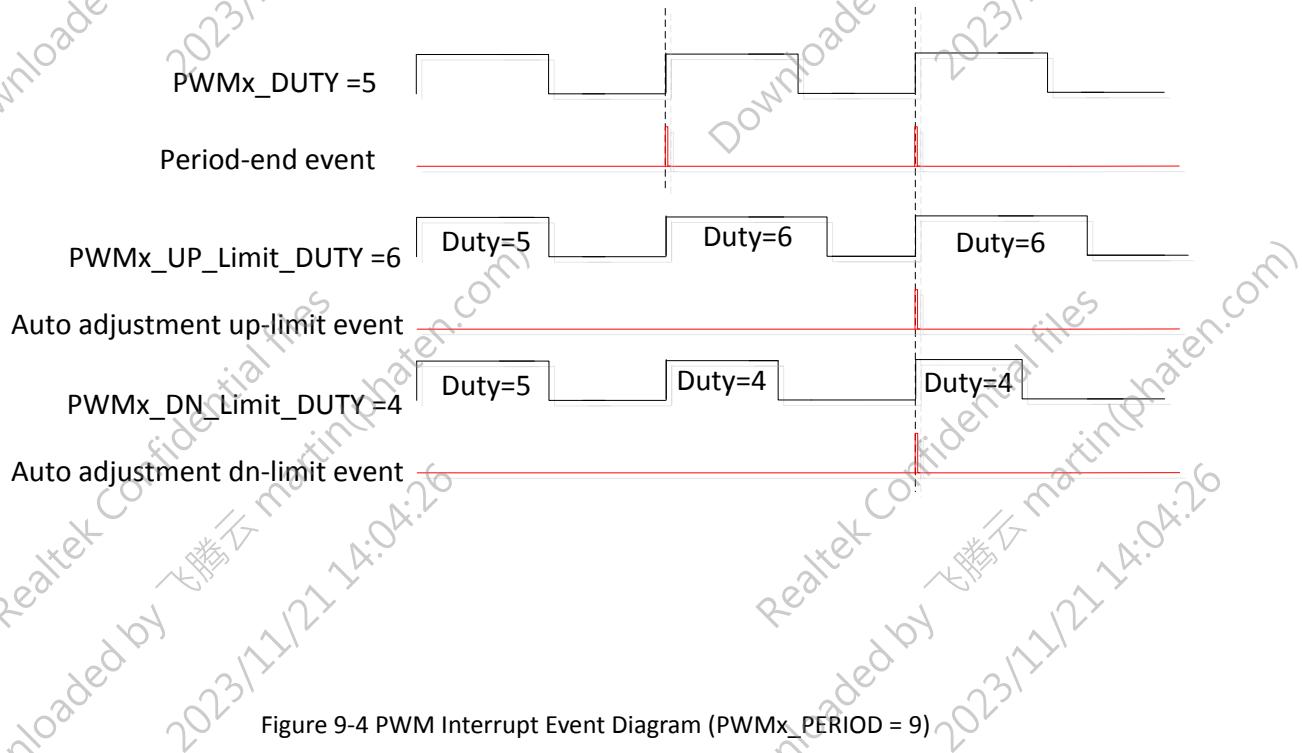


Figure 9-4 PWM Interrupt Event Diagram (PWMx_PERIOD = 9)

9.3 Control Registers

9.3.1 PWM_EN_STS

- Name: PWM enable status Register, which $x = 0 \sim 7$
- Width: 32bit
- Initial Value: 0x0000

REG 9-1 (Offset 0000h) PWM_EN_STS

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7:0	R	0	PWMx_EN_STS	0: the specific PWM is Disable 1: the specific PWM is Enable

9.3.2 PWM_EN

- Name: PWM enable Register, which $x = 0 \sim 7$
- Width: 32bit
- Initial Value: 0x0000

REG 9-2 (Offset 0004h) PWM_EN

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7:0	WO	0	PWM x _EN	When wrote 0: No operation 1: the specific PWM is enabled and the specific bit of REG_PWM_EN_STS is ONE.

9.3.3 PWM_DIS

- Name: PWM disable Register, which $x = 0 \sim 7$
- Width: 32bit
- Initial Value: 0x0000

REG 9-3 (Offset 0008h) PWM_DIS

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7:0	WO	0	PWM x _DIS	When wrote 0: No operation 1: the controlled PWM is disabled and the specific bit of REG_PWM_EN_STS is ZERO.

9.3.4 PWM_INT_STATUS

- Name: PWM Interrupt Status Register
- Width: 32bit
- Initial Value: 0x0000

REG 9-4 (Offset 000Ch) PWM_INT_STATUS

Bit	Access	INI	Symbol	Description
31	R/WC	0	PERIOD_END_IS7	To indicate the interrupt pending status for PWM7 period end. Write 1 clear.
30	R/WC	0	PERIOD_END_IS6	To indicate the interrupt pending status for PWM6 period end. Write 1 clear.

29	R/WC	0	PERIOD-END-IS5	To indicate the interrupt pending status for PWM5 period end. Write 1 clear.
28	R/WC	0	PERIOD-END-IS4	To indicate the interrupt pending status for PWM4 period end. Write 1 clear.
27	R/WC	0	PERIOD-END-IS3	To indicate the interrupt pending status for PWM3 period end. Write 1 clear.
26	R/WC	0	PERIOD-END-IS2	To indicate the interrupt pending status for PWM2 period end. Write 1 clear.
25	R/WC	0	PERIOD-END-IS1	To indicate the interrupt pending status for PWM1 period end. Write 1 clear.
24	R/WC	0	PERIOD-END-ISO	To indicate the interrupt pending status for PWM0 period end. Write 1 clear.
23	R/WC	0	DUTY-ADJ-UP-IS7	To indicate the interrupt pending status for PWM7 duty auto-adjustment reaches the up-limit. Write 1 clear.
22	R/WC	0	DUTY-ADJ-UP-IS6	To indicate the interrupt pending status for PWM6 duty auto-adjustment reaches the up-limit. Write 1 clear.
21	R/WC	0	DUTY-ADJ-UP-IS5	To indicate the interrupt pending status for PWM5 duty auto-adjustment reaches the up-limit. Write 1 clear.
20	R/WC	0	DUTY-ADJ-UP-IS4	To indicate the interrupt pending status for PWM4 duty auto-adjustment reaches the up-limit. Write 1 clear.
19	R/WC	0	DUTY-ADJ-UP-IS3	To indicate the interrupt pending status for PWM3 duty auto-adjustment reaches the up-limit. Write 1 clear.
18	R/WC	0	DUTY-ADJ-UP-IS2	To indicate the interrupt pending status for PWM2 duty auto-adjustment reaches the up-limit. Write 1 clear.
17	R/WC	0	DUTY-ADJ-UP-IS1	To indicate the interrupt pending status for PWM1 duty auto-adjustment reaches the up-limit. Write 1 clear.
16	R/WC	0	DUTY-ADJ-UP-ISO	To indicate the interrupt pending status for PWM0 duty auto-adjustment reaches the up-limit. Write 1 clear.
15:8	R	0	RSVD	
7	R/WC	0	DUTY-ADJ-DN-IS7	To indicate the interrupt pending status for PWM7 duty auto-adjustment reaches the down-limit. Write 1 clear.
6	R/WC	0	DUTY-ADJ-DN-IS6	To indicate the interrupt pending status for PWM6 duty auto-adjustment reaches the down-limit. Write 1 clear.
5	R/WC	0	DUTY-ADJ-DN-IS5	To indicate the interrupt pending status for PWM5 duty auto-adjustment reaches the down-limit. Write 1 clear.
4	R/WC	0	DUTY-ADJ-DN-IS4	To indicate the interrupt pending status for PWM4 duty auto-adjustment reaches the down-limit. Write 1 clear.
3	R/WC	0	DUTY-ADJ-DN-IS3	To indicate the interrupt pending status for PWM3 duty auto-adjustment reaches the down-limit. Write 1 clear.
2	R/WC	0	DUTY-ADJ-DN-IS2	To indicate the interrupt pending status for PWM2 duty auto-adjustment reaches the down-limit. Write 1 clear.

1	R/WC	0	DUTY_ADJ_DN_IS1	To indicate the interrupt pending status for PWM1 duty auto-adjustment reaches the down-limit. Write 1 clear.
0	R/WC	0	DUTY_ADJ_DN_IS0	To indicate the interrupt pending status for PWM0 duty auto-adjustment reaches the down-limit. Write 1 clear.

9.3.5 PWM_INDREAD_IDX

- Name: PWM Index of Indirect Read Register
- Width: 32bit
- Initial Value: 0x0000

REG 9-5 (Offset 0010h) PWM_INDREAD_IDX

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7	W1S	--	POOL	Set this bit to enable indirect read current value of timer selected by PWM_INDREAD_IDX. This bit is cleared by HW while finishing read and indicates REG_PWM_INDREAD_DUTY is ready.
6:5	R/W	0	RSVD	Reserve function
4	R/W	0	SYNC_MODE	Mode selection. 0: Freq. of APB clock >> Freq. of Timer clock (about 10 times) 1: Freq. of Timer clock >= Freq. of APB clock
3	R/W	0	RSVD	Reserve function
2:0	R/W	0	PWM_INDREAD_IDX	This field is used to assign the PWM index (0 ~ 7) for the auto adjusted duty size indirect reading.

9.3.6 PWM_INDREAD_DUTY

- Name: PWM Auto Adjusted Duty Indirect Read Register
- Width: 32bit
- Initial Value: 0x0000

REG 9-6 (Offset 0014h) PWM_INDREAD_DUTY

Bit	Access	INI	Symbol	Description
31:12	R	0	RSVD	
11:0	R/W	0	PWM_INDREAD_DUTY	This field is used to read the auto adjusted duty size of the PWM which is assigned by the PWM_INDREAD_IDX. The SW should assign the PWM to read by write the PWM_INDREAD_IDX and then read this field to get the current duty size of the specified PWM.

9.3.7 PWMx_CTRL

- Name: PWMx Control Register, which x = 0 ~ 7
- Width: 32bit
- Initial Value: 0x0000

REG 9-7 (Offset 000h + (0x20h * (x+1))) PWMx_CTRL

Bit	Access	INI	Symbol	Description
31	R/W	0	PWMx_CTRL_SET	0: 1: SW can change setting only at PWMx_CTRL_SET = 0, and set PWMx_CTRL_SET = 1 after changing PWM Ctrl. HW clear PWMx_CTRL_SET after changing PWM Ctrl.
30	R/W	0	PWMx_PAUS	0: Normal running 1: To paus the PWM out at current PWM period end. Since the PWM output is not stopped immediately, the software can poll the PWMx_RUNSTS to know the PWM out is still running or is paused.
29	R	0	PWMx_RUNSTS	The PWM output running status: 0: PWM output is paused 1: the PWM output is running.
28	R/W	0	PWMx_PERIOD_INT	To control the PWM period end interrupt enable (1) or disable (0). When the PWM period end interrupt is enabled, the PWM HW will assert the interrupt on every PWM period end time. This interrupt status can be read from REG_PWM_INT_STATUS.
27:16	R	--	PWMx_ADJ_DUTY	The current duty size which is adjusted by the duty auto-adjustment HW. The SW can read this field to know the current duty size. However, the latest duty size may not be sync to this field on time or the value may transition while the SW is reading this field. So use in-direct read method to read the duty size is more save and is suggested.
15:12	R/W	0	PWMx_GT_SEL	0: GTIMER 0 1: GTIMER 1 ... 7: GTIMER 7 8: sclk
11:0	R/W	0	PWMx_DUTY	The on-duty duration of PWM pulse. The time unit is configured from GTIMER specified by PWMx_GT_SEL field. It can be written at any time, but HW change setting only at PWM enable and the end of period.

9.3.8 PERI_PWMx_TIM_CTRL

- Name: PWMx Timing Control Register, which x = 0 ~ 7
- Width: 32bit
- Initial Value: 0x0000

REG 9-8 (Offset 004h + (0x20h * (x+1))) PWMx_TIM_CTRL

Bit	Access	INI	Symbol	Description
31:28	R/W	0	RSVD	Reserve function
27:16	R/W	0	PWMx_DUTY_START	The start of ON duration of PWM pulse in the period. The time unit is the same as PWM period. It can be written at any time, but HW change setting only at PWM enable and the end of period.
15:12	R/W	0	RSVD	Reserve function
11:0	R/W	0	PWMx_PERIOD	The period of PWM pulse. The time unit is configured from GTIMER specified by PWMx_GT_SEL field. It can be written at any time, but HW change setting only at PWM enable and the end of period.

9.3.9 PWMx_DUTY_AUTO_ADJ_CTRL

- Name: PWMx Duty Auto Adjustment Control Register, which x = 0 ~ 7
- Width: 32bit
- Initial Value: 0x0000

REG 9-9 (Offset 008h + (0x20h * (x+1))) PWMx_DUTY_AUTO_ADJ_CTRL

Bit	Access	INI	Symbol	Description
31	R/W	0	DUTYx_ADJ_EN	To enable the Duty Ato Adjustment. 0: Disable 1: Enable
30	R/W	0	DUTYx_ADJ_DIR	To set the Duty Ato Adjustment direction. 0: Decrease Duty 1: Increase Duty
29	R/W	0	DUTYx_ADJ_LOOP	To enable the Duty Auto Adjustment Loop mode. 0: Disable 1: Enable If the Loop Mode is enabled, reverse the Duty Auto Adjustment direction when the adjusted Duty reach the Up Limit or the Down Limit.
28	R/W	0	DUTYx_ADJ_UP_INT	To enable the Interrupt of the Duty Auto Adjustment. 0: Disable 1: Enable If the Interrupt is enabled, issue an interrupt when the Adjusted Duty reach the Up Limit.
27	R/W	0	DUTYx_ADJ_DN_INT	To enable the Interrupt of the Duty Auto Adjustment. 0: Disable 1: Enable If the Interrupt is enabled, issue an interrupt when the Adjusted Duty reach the Down Limit.
26:24	R/W	0	RSVD	Reserve function
23:22	R/W	0	RSVD	Reserve function
21:12	R/W	0	DUTYx_ADJ_INC_STEP	The Duty Increasing Step size of the Duty Auto Adjustment.
11:10	R/W	0	RSVD	Reserve function
9:0	R/W	0	DUTYx_ADJ_DEC_STEP	The Duty Decreasing Step size of the Duty Auto Adjustment.

9.3.10 PWMx_DUTY_AUTO_ADJ_LIMIT

- Name: PWMx Duty Auto Adjustment Limit Register, which x = 0 ~ 7
- Width: 32bit
- Initial Value: 0x0000

REG 9-10(Offset 00Ch + (0x20h * (x+1))) PWMx_DUTY_AUTO_ADJ_LIMIT

Bit	Access	INI	Symbol	Description
31:28	R/W	0	RSVD	Reserve function
27:16	R/W	0	DUTYx_ADJ_UP_LIM	The Up Limit of the Adjusted Duty.
15:12	R/W	0	RSVD	Reserve function
11:0	R/W	0	DUTYx_ADJ_DWN_LIM	The Down Limit of the Adjusted Duty.

9.3.11 PWM_DUTY_AUTO_ADJ_CYCLE

- Name: PWMx Duty Auto Adjustment Cycle Count Register, which x = 0 ~ 7
- Width: 32bit
- Initial Value: 0x0000

REG 9-11 (Offset 010h + (0x20h * (x+1))) PWMx_DUTY_AUTO_ADJ_CYCLE

Bit	Access	INI	Symbol	Description
31:12	R/W	0	RSVD	Reserve function
11:0	R/W	--	DUTYx_ADJ_CYCLE_CNT	The Cycle Count of the Duty Auto Adjustment. The Duty size will be increased/decreased with a step size every Cycle Count of PWM period.

10 Universal Asynchronous Receiver/ Transmitter (UART)

10.1 Overview

10.1.1 Application Scenario

The Low Power UART is specified for low power consumption and low data rate application. The power consumption is the most important consideration, so a hardware match filter is designed to filet RX data, and then wake up the CPU from sleep mode when the RX data is matched with CPU wake condition. By this way, the CPU will be waked up only when needed.

10.1.2 Features

- UART (RS232 Standard) Serial Data Format
- Support up to baud rate 115200 with 4 MHz clock source
- 16 bytes Transmit Data FIFO
- 32 bytes Receive Data FIFO
- Programmable Receive Data FIFO Trigger Level
- Programmable RX Filter (A RX data match Filter to check RX data and then wake up CPU from sleep mode.)
- DMAC interface (DMA data moving support to save MCU loading.)
- Auto flow control.

10.1.3 Architecture

The UART interface is a standard 4-wire interface with RX, TX, CTS, and RTS. Users basically can set TX data or get RX data from Transmitter Holding Register/Receiver Buffer Register. To set or get more information of TX/RX FIFO via accessing FIFO Control registers. In order to generate the desired baud rate and data format, users can access configuration registers which are related to line control information and Baud rate setting parameters. There are also DMAC channels for UART TX/RX mode transfer. A simplified block diagram of the component is illustrated in Figure 10-1.

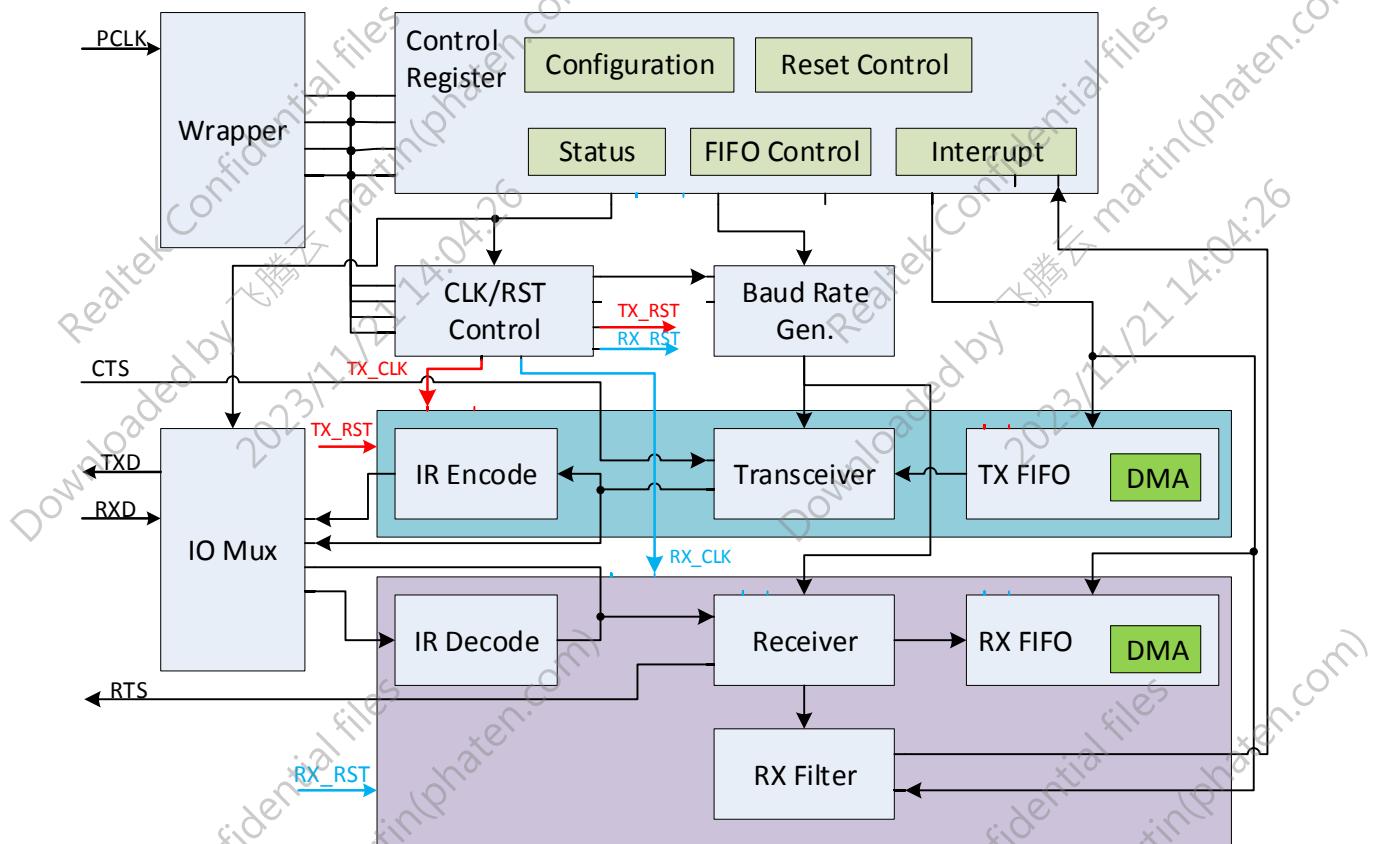


Figure 10-1 UART Block Diagram

10.1.3.1 Functional Block Diagram with Clock Domain

- **Wrapper:** Takes the bus interface signals and translates them into a common generic interface that allows the register file to be bus protocol-agnostic.
- **Control register:** Contains configuration registers and is the interface with software.
- **Interrupt controller:** Generates the raw interrupt and interrupt flags, allowing them to be set and cleared.
- **Baud Rate Generate:** Generates the baud rate for UART protocol.
- **IR Encode/Decode:** SIR encoder/decoder of UART Tx/Rx data.
- **TX/RX FIFO:** Asynchronous FIFO.
- **DMA interface:** Generates the handshaking signals to the central DMA controller in order to automate the data transfer without CPU intervention.
- **RX Filter:** Filter to check Rx Data 1st byte/2nd byte whether it matches pattern.

10.2 Functional Description

Ameba-Z II's UART basically adopts general UART design and provides extended features for reducing CPU computing and power consumption. Several different operation types could be applied for different application requirements.

10.2.1 Overview

The UART is serial data communication which is the data can be transferred through a wire in a bit-by-bit form, and it uses two wires to transmit data between devices. Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART. UART transmit data is asynchronous, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART, there is a communication protocol which adds three bits like start, stop and parity to help transmitting data. The connection is illustrated in below:

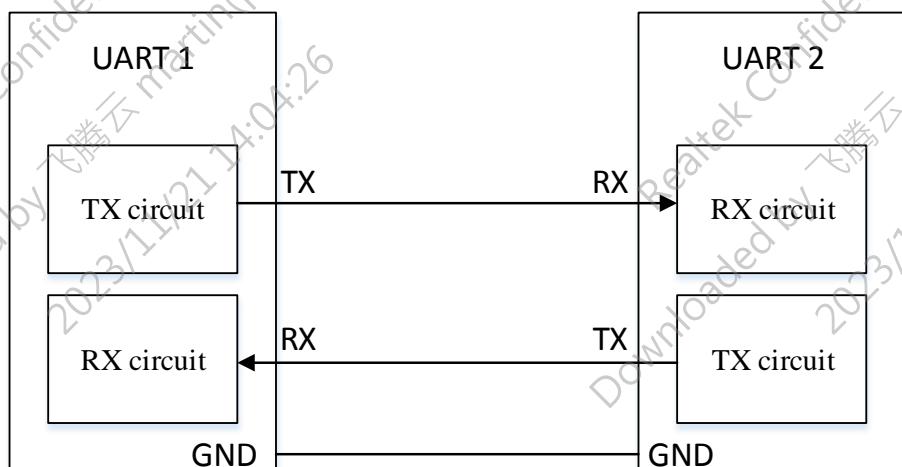


Figure 10-2 UART Wire Connection

10.2.2 UART Terminology

The following terms are used throughout this manual and are defined as follows:

10.2.2.1 UART Data Format

The UART transmits data is organized into packets. Each packet at least contains 1 start bit, data bits and stop bit. As for the parity bit, it is optional.

- **Start Bit:** The UART data transmission line is normally held at a high level voltage when it's not transmitting data. To start the transfer of data, the transmitting UART pulls the transmission line from high to low for one bit period. When the receiving UART detects the high to low voltage transition, it begins reading the bits in the data frame.
- **Data Bits:** The data bits include the real data being conveyed from the sender to receiver. In Ameba-Z II's UART, user can set 7/8 bits word length. Generally, the LSB of the data to be transmitted first; The MSB of the data to be transmitted last.
- **Parity Bit:** Because bits can be changed by electromagnetic radiation, mismatched baud rates, or long distance data transfers, the parity bit let the receiver to ensure whether the collected data is right or not. The receiving UART reads the data bits, it counts the number of bits with a value of 1 and checks if the total is an even or odd number. When the parity bit matches the data, the UART knows that the transmission was correct. Actually, this bit is not widely used so it's optional. In Ameba-Z II's UART, user can set no parity/odd,even parity.
- **Stop Bit:** The stop bit indicates the end of the data packet. The transmitting UART drives the data transmission line from a low voltage to a high voltage. In Ameba-Z II's UART, user can set 1/2 stop bits.

The data format is illustrated in below:

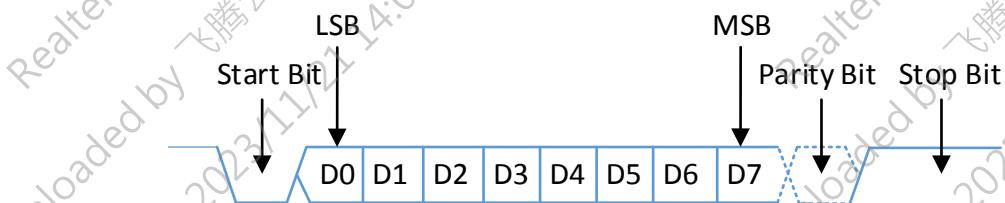


Figure 10-3 UART Data Format

10.2.3 Baud Rate Calculation

The baud rate calculation of the receiver and transmitter (Rx and Tx) is based on below formula:

$$\text{Average Baud Rate} = \frac{\text{clock_freq}}{\left(\text{ovsr}[3:0] + 5 + \frac{\text{ovsr}_{\text{adj}}[0] + \text{ovsr}_{\text{adj}}[1] + \dots + \text{ovsr}_{\text{adj}}[10]}{11} \right) * (\text{divisor}[15:0])}$$

The parameters of this fractional baud rate ($\text{ovsr}[3:0]$, $\text{ovsr}_{\text{adj}}[10:0]$ and $\text{divisor}[15:0]$) are configured depending on IP clock frequency and expected baud rate. PCLK is using CPU clock. AS for SCLK, assume that we support the baud rate 115200 with 3% tolerance, and use the clock frequency value is times of 1200. If we want to make the SCLK Freq. is close to 4 MHz,

then the suggestion Freq. of the SCLK is 3993600 Hz. Below table shows that baud rate error calculation for SCLK Freq. is 3993600 with 1.5% offset.

Table 10-1 Baud Rate Error Calculation

baud rate	ovsr	adj	div	err	act baud(parity)	Err	start_bit_err(%)	freq_offset(%)	Jitter(%)	ovsr_jitter	word_freq_offset(%)
4800	13	0	64	0	4800	0	0.240384615	1.5	1.5	0	33.24038462
9600	13	0	32	0	9600	0	0.480769231	1.5	1.5	0	33.48076923
19200	13	0	16	0	19200	0	0.961538462	1.5	1.5	0	33.96153846
38400	13	0	8	0	38400	0	1.923076923	1.5	1.5	0	34.92307692
76800	13	0	4	0	76800	0	3.846153846	1.5	1.5	0	36.84615385
115200	17	3	2	1.26E-16	115421.9653	0.001927	5.769230769	1.5	1.5	5.882352941	44.65158371

Note: The word_freq_offset must be lower than 50%. If the value is lower than 40%, it is safer.

However, the actual SCLK Freq. is 4 MHz with 2% frequency offset. Below table shows a baud rate error calculation for this SCLK Freq.

Table 10-2 Error Rate of Baud Rate

Desired Baud Rate	Actual Baud Rate	Error(%)	Desired Baud Rate	Actual Baud Rate	Error(%)
110	110.0533759	0.048523534	380400	380952.381	0.145210555
300	300.120048	0.040016006	460800	460732.9843	0.014543339
600	600.240096	0.040016006	500000	500000	0
1200	1200.480192	0.040016006	921600	922431.8658	0.090263219
2400	2400.960384	0.040016006	1000000	1000000	0
4800	4801.920768	0.040016006	1382400	1383647.799	0.090263219
9600	9603.841537	0.040016006	1444400	1452145.215	0.536223658
14400	14414.41441	0.1001001	1500000	1506849.315	0.456621005
19200	19230.76923	0.16025641	1843200	1856540.084	0.723745898
28800	28860.02886	0.208433542	2000000	2000000	0
38400	38461.53846	0.16025641	2100000	2105263.158	0.250626566
57600	57720.05772	0.208433542	2764800	2784810.127	0.723745898
76800	76923.07692	0.16025641	3000000	3013698.63	0.456621005
115200	115243.583	0.037832489	3250000	3283582.09	1.033295063
128000	128205.1282	0.16025641	3692300	3728813.559	0.988910959
153600	153846.1538	0.16025641	3750000	3793103.448	1.149425287
230400	231092.437	0.300536881	4000000	4000000	0

10.2.4 Auto Flow Control

Use a flow control mechanism which is a part of the RS232 protocol so that communicate with each other device over UART without the risk of losing data. This mechanism needs 2 extra wires which are called RTS(Request to Send) and CTS(Clear to Send). RTS on one device is

connected to CTS on the remote device and vice versa (Reference Figure 10-4).

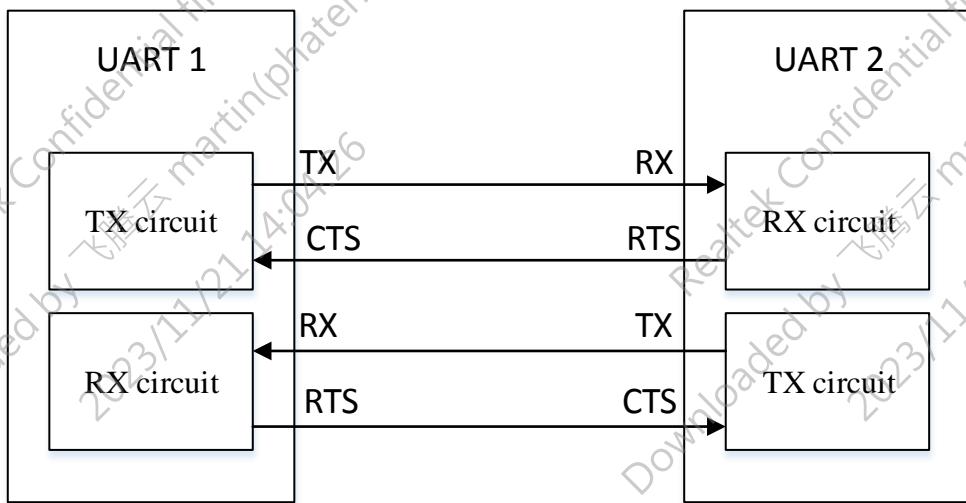


Figure 10-4 Hardware Flow Control Between 2 UARTs

When auto-flow control mechanism is enabled, each device will use its RTS to output if it is ready to accept new data and read CTS to see if it is allowed to send data to the other device. Keep the RTS line asserted(tied low) as long as a device is ready to accept more data. When the receive buffer is full, RTS is deasserted to inform the other device to stop transmission at the end of current transmission. If CTS is asserted(tied low), then the next data is transmitted, else the transmission doesn't occur. If input CTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

10.2.5 RX Filter

For some application, the system can be waked up from sleep mode by receiving a packet with special characters ahead. To reduce the power consumption of the system when it is in sleep mode, the RX filter hardware is designed to check the first 1 or 2 bytes of a packet from the UART receiving. So the CPU no needs to be waked up to check every received UART byte. The CPU will be waked up only when an ‘interested’ packet is received. Fig. 2-4 shows the block diagram of the RX filter.

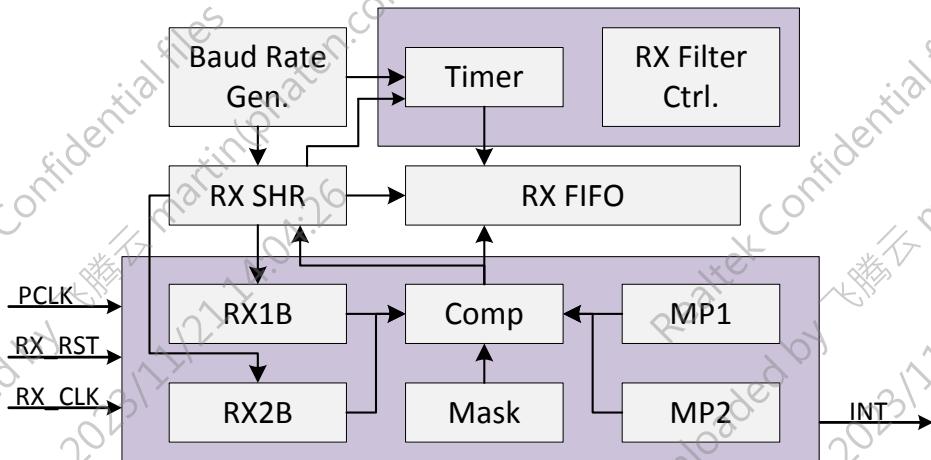


Figure 10-5 UART Rx Filter Block Diagram

The mechanism of the RX filter is:

- The RX filter will be reset as initial state by the RX_RST signal. In the initial state, the 1st received byte will be put into RX1B and the 2nd byte will be put into RX2B. The first 2 received bytes will be push into the RX FIFO also. The software will set the pattern for the received data checking in MP1 and MP2. The received bytes in RX1B and RX2B will do a “AND” operation with the value in MASK optionally. And then use the result to compare with MP1 and MP2. If the result of the comparing is “matched” then all following received data will be push into RX FIFO and an interrupt signal will be issued to wake up the CPU. If the compare result is “not match”, the RX FIFO will be flushed and all following received bytes will be dropped till the receiving is idle (no new RX data for a while).
- The rule of received data comparing is configurable. It can just compare the first received byte or compare first 2 received bytes. The rule for data comparing can be:
 - AND: the first byte (with mask) is equal to MP1; both first 2 bytes (with mask) are equal to MP1 and MP2.
 - OR: the first byte (with mask) is equal to MP1 or MP2.
 - XOR: the first byte (with mask) is not equal to MP1 neither equal to MP2; the first byte (with mask) is not equal to MP1 and the 2nd byte is not equal to MP2.
- Texas Instruments Serial Protox:
- The RX idle detection mechanism is used to check whether the receiving is in idle state. This mechanism uses a timer to monitor new data receiving. This timer will reload the initial value whenever a new byte is received. If the timeout occurred (timer value countdown to 0), the RX filter will be reset to the initial state and restart the first received 1 or 2 bytes checking.

10.3 Control Registers

10.3.1 DLLR

- Name: Divisor Latch LS Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is the baud rate divisor[7:0], can be programmed only when the DLAB bit of LCR bit [7] = 1.

$$\text{AverageBaudRate} = \frac{\text{clock_freq}}{(ovsr[3:0]+5+\frac{ovsr_adj[0]+ovsr_adj[1]+\dots+ovsr_adj[10]}{11})*(divisor[15:0])}$$

REG 10-3 (Offset 0000h) DLLR

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7:0	R/W	0	DLL	Divisor [7:0]; accessible when DLAB = 1

10.3.2 DLMR

- Name: Divisor Latch MS Register
- Width: 32bit
- Initial Value: 0x0010
- Note: This register is the baud rate divisor[15:8] , can be programmed only when the DLAB bit of LCR bit [7] = 1.

$$\text{AverageBaudRate} = \frac{\text{clock_freq}}{(ovsr[3:0]+5+\frac{ovsr_adj[0]+ovsr_adj[1]+\dots+ovsr_adj[10]}{11})*(divisor[15:0])}$$

REG 10-4 (Offset 0004h) DLMR

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7:0	R/W	0	DLM	Divisor [15:8]; accessible when DLAB = 1

10.3.3 IER

- Name: IER Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register allows enabling and disabling interrupt generation by the UART. It can be accessed only when the DLAB bit of LCR bit [7] = 0. This register enables the four types of UART interrupts. Each interrupt can individually activate the interrupt (INTR) output signal. It is possible to totally disable the interrupt system by resetting bit 0 through 3 of the Interrupt Enable Register (IER). Similarly, setting bits of the IER register to a logic 1 enables the selected interrupt(s). Disabling an interrupt prevents it from being indicated as active in the IIR and from activating the INTR output signal. All other system functions operate in their normal manner, including the setting of the Line Status and Modem Status Registers.

REG 10-5 (Offset 0004h) IER

Bit	Access	INI	Symbol	Description
31:4	R	0	RSVD	
3	R/W	0	EDSSI	Enable Modem Status Interrupt (EDSSI)(modem status transition) ● 0: disabled ● 1: enabled
2	R/W	0	ELSI	Enable Receiver Line Status Interrupt (ELSI)(receiver line status) ● 0: disabled ● 1: enabled
1	R/W	0	ETBEI	Enable Transmitter FIFO Empty interrupt (ETBEI)(tx fifo empty) ● 0: disabled ● 1: enabled
0	R/W	0	ERBI	Enable Received Data Available Interrupt (ERBI)(rx trigger) ● 0: disabled ● 1: enabled

10.3.4 IIR

- Name: Interrupt Identification Register
- Width: 32bit
- Initial Value: 0x0001
- Note: This register enables the programmer to retrieve what is the current highest priority pending interrupt. In order to provide minimum software overhead during data character transfers, the UART prioritizes interrupts into four levels and records these in the interrupt Identification Register. The four levels of interrupt conditions in order of priority are Receiver Line Status; Received Data Ready (Trigger or Timeout); Transmitter FIFO Empty;

and Modem Status. When the CPU accesses the IIR, the UART freezes all interrupts and indicates the highest priority pending interrupt to the CPU. While this CPU access is occurring, the UART record new interrupts, but does not change its current indication until the access is complete.

REG 10-6 (Offset 0008h) IIR

Bit	Access	INI	Symbol	Description
31:4	R	0	RSVD	
3:1	R	0	INT_ID[2:0]	<p>The two bits of the IIR(Bit1 and Bit2) are used to identify the highest priority interrupt pending as indicated in the following table.</p> <p>Bit3: In the FIFO mode this bit is set along with bit 2 when a timeout interrupt is pending.</p> <p>Bit3~Bit1 displays the list of possible interrupts along with the bits they enable, priority, and their source and reset control.</p> <ul style="list-style-type: none"> ● 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 Modem status register ● 3'b001: <ul style="list-style-type: none"> ■ Interrupt Priority: 3rd priority ■ Interrupt Type: TXFIFO empty ■ Interrupt source: TXFIFO empty ■ Interrupt Reset Control: Writing to the TX FIFO(THR) or reading IIR (if source of interrupt) ● 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(depend on FCR[7:6]: rcvr_trig) <ul style="list-style-type: none"> ◆ 00: 1 ◆ 01: 8 ◆ 10: 16 ◆ 11: 28 ● 3'b011: <ul style="list-style-type: none"> ■ Interrupt Priority: 1st priority (int_3) ■ Interrupt Type: Receiver Line Status (read lsR) ■ Interrupt source: Parity, overrun or Framing errors or break interrupt ■ Interrupt Reset Control: Reading the Line Status Register(LSR) ● 3'b110: <ul style="list-style-type: none"> ■ Interrupt Priority: 2nd priority ■ Interrupt Type: Timeout Indication ■ Interrupt source: There's at least 1 character in the FIFO but no character has been input to the FIFO or

					read from it for the last 4 characters times. If there is no reading to the FIFO (through RBR) after this interrupt is triggered, it will be triggered again and again every 6 characters times till the FIFO is empty or another character is pushed into the FIFO.
0	R	1	INT_PEND		<ul style="list-style-type: none"> ■ Interrupt Reset Control: Reading Receiver Buffer Register (RBR) <p>Indicates that an interrupt is pending when it's logic "0". When it is "1", no interrupt is pending.</p> <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.

Table 10-7 Interrupt Control Functions

FIFO Mode Only	Interrupt Identification Register			Interrupt Set and Reset Functions				
Bit 3	Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control	
0	0	0	1	-	None	None	-	
0	1	1	0	Highest (3)	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register(LSR)	
0	1	0	0	Second (2)	Received Data Available	Trigger Level Reached	Reading the Receive FIFO to makes the FIFO level drops below the Trigger Level	
1	1	0	0	Second (2)	Character Timeout Indication	No Characters have been removed from or input to the receive FIFO during the last 4 characters times and there is at least 1 character in it during this time,	Reading the Receive FIFO (RBR)	
0	0	1	0	Third (1)	TXFIFO Empty	TX FIFO Empty	Reading the IIR Register (if source of interrupt) or writing into the TXFIFO	
0	0	0	0	Fourth (0)	Modem Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the Modem Status Register(MSR)	

10.3.5 FCR

- Name: FIFO Control Register
- Width: 32bit
- Initial Value: 0x0009
- Note: This register is used to enable the FIFOs, clear the TX/RX FIFOs, set the Receiver FIFO

Trigger level, and select the type of DMA signaling. This is a write only register at the same location as the IIR (the IIR is a read only register). This register allows selection of the FIFO trigger level (the number of bytes in FIFO required to enable the Received Data Available interrupt). In addition, the FIFOs can be cleared using this register.

REG 10-8 (Offset 0008h) FCR

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7:6	W	0	RXFIFO_TRIGGER_LEVEL	Define the 32-entries Receiver FIFO Interrupt trigger level 0~31 bytes <ul style="list-style-type: none"> ● 2'b00: 1 byte ● 2'b01: 8 bytes ● 2'b10: 16 bytes ● 2'b11: 28 bytes (for some device detect RTS de-assertion slower, reserve more RX FIFO space to prevent RX FIFO overflow)
5	W	0	TX_FIFO_LOW_LEVEL	Define the Transmission FIFO Low Water Level Interrupt trigger. <ul style="list-style-type: none"> ● 0: 4 byte ● 1: 8 bytes
4	R	0	RSVD	
3	W	1	DMA_MODE	Support DMA mode. (cooperate with DW DDMA in the data path) <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable
2	W	0	CLEAR_TXFIFO	Writing logic "1" to Bit 2 clears the Transmitter FIFO and resets its logic. The shift register is not cleared, i.e., transmitting of the current character continues. The 1 that is written to this bit position is self-clearing
1	W	0	CLEAR_RXFIFO	Writing logic "1" to Bit 1 clears the Receiver FIFO and resets its logic. But it doesn't clear the shift register, i.e. receiving of the current character continues. The 1 that is written to this bit position is self-clearing
0	W	1	EN_RXFIFO_ERR	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

10.3.6 LCR

- Name: Line Control Register
- Width: 32bit
- Initial Value: 0x0000
- Note: The system programmer specifies the format of the asynchronous data communications exchange via the Line Control Register (LCR).

REG 10-9 (Offset 000Ch) LCR

Bit	Access	INI	Symbol	Description															
31:8	R	0	RSVD																
7	R/W	0	DLAB	<p>Divisor Latch Access bit</p> <ul style="list-style-type: none"> ● 0: The divisor latches cannot be accessed ● 1: The divisor latches can be accessed ● Note: DLL/DLM only can be access when dlab bit = 1 IER only can be access when dlab bit = 0 THR/RBR doesn't care about dlab bit value. 															
6	R/W	0	BREAK_CTRL	<p>Break Control bit</p> <ul style="list-style-type: none"> ● 0: Break is disabled. ● 1: The serial out is forced into logic '0' (break state). Break control bit causes a break condition to be transmitted to the receiving UART. When it is set to logic 1, the serial output (Sout) is forced to the Spacing (logic 0) state. The break is disabled by setting bit 6 to logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic. 															
5	R/W	0	STICK_PARITY_EN	<p>Stick Parity enable control.</p> <ul style="list-style-type: none"> ● 0: Disable Stick Parity. ● 1: Enable Stick Parity. <p>If the stick parity is enabled, the parity bit is controlled by the LCR bit[4].</p> <table border="1"> <thead> <tr> <th>LCR[5]</th><th>LCR[4]</th><th>Parity</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>Odd Parity</td></tr> <tr> <td>0</td><td>1</td><td>Even Parity</td></tr> <tr> <td>1</td><td>0</td><td>Stick Parity as 0</td></tr> <tr> <td>1</td><td>1</td><td>Stick Parity as 1</td></tr> </tbody> </table>	LCR[5]	LCR[4]	Parity	0	0	Odd Parity	0	1	Even Parity	1	0	Stick Parity as 0	1	1	Stick Parity as 1
LCR[5]	LCR[4]	Parity																	
0	0	Odd Parity																	
0	1	Even Parity																	
1	0	Stick Parity as 0																	
1	1	Stick Parity as 1																	
4	R/W	0	EVEN_PARITY_SEL	<p>Even Parity select</p> <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 "1" is transmitted in each word. 															
3	R/W	0	PARITY_EN	<p>Parity Enable</p> <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	R/W	0	STB	<p>This bit specifies the number of Stop bits transmitted and received in each serial character.</p> <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	R	1	RSVD																
0	R/W	1	WLS0	<p>Word length selection 0</p> <ul style="list-style-type: none"> ● 0: Data is 7 bit word length. ● 1: Data is 8 bit word length. 															

10.3.7 MCR

- Name: Modem Control Register
- Width: 32bit
- Initial Value: 0x0000
- Note: The Modem Control Register allows transferring control signal to a modem connected to the UART. This register controls the interface with the modem or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register are described below.

REG 10-10 (Offset 0010h) MCR

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
7	R/W	0	SW_CTS	<p>Software controlled CTS. The software can use this bit to pause the UART transmission, just like the HW flow control. This is useful for software to implement XOn/XOff SW flow control. This bit setting will effect the CTS flow-control:</p> $\text{CTS} = \text{cts_en} ? (\text{sw_cts} \text{CTS_input}) : \text{sw_cts}$
6	R/W	0	RTS_EN	<p>RTS flow control enable (RTSE) This Bit (RTSE) is the auto RTS flow control enables. When set (1), the auto RTS flow control as described in the detailed description is enabled.</p> <p>Note: Auto-RTS_ data flow control originates in the receiver timing and control block and is linked to the programmed receiver FIFO trigger level. When the receiver FIFO level reaches a trigger level, RTS_ is de-asserted. The sending communication element may send an additional byte after the trigger level is reached (assuming the sending communication element has another byte to send) because it may not recognize the de-assertion of RTS_ until after it has begun sending the additional byte. RTS_ is automatically reasserted once the Receive FIFO is emptied.</p>
5	R/W	0	CTS_EN	<p>CTS flow control enable (CTSE) This Bit (CTSE) is the auto CTS flow control enable. When set (1), the auto CTS flow control as described in the detailed description is enabled.</p> <p>Note: Auto-CTS_: The Transmitter circuitry checks CTS_ before sending the next data byte. When CTS_ is active, it sends the next byte. To stop the transmitter from sending the following byte, CTS_ must be released before the middle of the last stop bit that is currently being sent. The auto-CTS_ function reduces interrupts to the host system. When flow control is enabled, CTS_ level changes do not trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS_, the transmitter sends any data present in the transmitter FIFO and a receiver overrun error may result.</p> <p>Note: With auto-CTS_, the CTS_ input must be active before the</p>

				<p>transmitter FIFO can emit data. With auto-RTS_, RTS_ becomes active when the receiver needs more data and notifies the sending serial device. When RTS_ is connected to CTS_, data transmission does not occur unless the receiver FIFO has space for the data; thus, overrun errors are eliminated with the auto flow control enabled. If not, overrun errors occurs when the transmit data rate exceeds the receiver FIFO read latency.</p> <p>Note: In the diagnostic mode, data that is transmitted is immediately received. This allows the CPU to verify the transmitting and receive data paths. The receiver and transmitter interrupts are fully operational. The modem control interrupts are also operational, but the modem control interrupt's sources are now the lower four bits of the MCR instead of four the four modem control inputs. All interrupts are still controlled by the IER</p>
4	R/W	0	LOOPBACK_EN	<p>LoopBack mode</p> <ul style="list-style-type: none"> ● 0: Normal operation ● 1: Loopback mode <p>This bit provides a local loopback feature for diagnostic testing of the UART. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is “looped back” into the Receiver Shift Register input; the four MODEM control inputs (DSR_, CTS_, RI_ and DCD_) are disconnected; and the four MODEM Control output (DTR_, RTS_, OUT1_, and OUT2_) are internally connected to the four MODEM Control inputs, and the MODEM Control output pins are forced to their inactive state (high). In the loop-back mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit-and-received data paths of the UART.</p> <p>In the loopback mode, the receiver and transmitter interrupts are fully operational. Their sources are external to the part. The MODEM Control interrupts are also operational, but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register.</p>
3	R/W	0	OUT2	<p>Output2</p> <p>This bit controls the output 2 (OUT2_) signal, which is an auxiliary user-designated output. Bit3 affects the OUT2_ in a manner identical to that described above for bit 0.</p> <p>In loopback mode, connected to Data Carrier Detect (DCD)</p>
2	R/W	0	OUT1	<p>Output1</p> <p>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 above for bit 0.</p> <p>In loopback mode, connected Ring Indicator (RI) signal input.</p>
1	R/W	0	RTS	<p>Request to Send (RTS) signal control</p> <ul style="list-style-type: none"> ● 0: RTS is logic 1 ● 1: RTS is logic 0 <p>This bit controls the Request to Send (RTS) output. The RTS output is controlled as following equation:</p> $\text{RTS_output} = \text{rts_en} ? (\sim \text{rts} \text{FIFO_FlowCtrl}) : \sim \text{rts}$

0	R/W	0	DTR	<p>Data Terminal Ready (DTR) signal control</p> <ul style="list-style-type: none"> ● 0: DTR is logic 1 ● 1: DTR is logic 0 <p>This bit controls the Data Terminal Ready (DTR_) output. When bit 0 is set to logic 1, the DTR_ output is forced to logic 0. When bit0 reset to logic 0, the DTR_ output is forced to logic 1.</p>
---	-----	---	-----	--

10.3.8 LSR

- Name: Line Status Register
- Width: 32bit
- Initial Value: 0x0060
- Note: This register provides status information to the CPU concerning the data transfer.

Bit 1 ~ Bit 4 are the error conditions that produce a Receiver Line Status Interrupt whenever any of the corresponding conditions are detected and the interrupt is enabled. The Line Status Register is intended for read operation only. Writing to this register is not recommended as this operation is only used for factory testing. In the FIFO mode the software must load a data byte in the RX FIFO via loopback mode in order to write to LSR2 – LSR4, LSR0 and LSR7 can't be written to in FIFO mode.

REG 10-11 (Offset 0014h) LSR

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7	R	0	RXFIFO_ERR	<p>UART Rx FIFO Error</p> <ul style="list-style-type: none"> ● 1: In the FIFO mode, this bit (LSR bit7) is set when there is at least one parity error, framing error or break indication in the FIFO. LSR7 is clear when the CPU reads the LSR, if there are no subsequent errors in the FIFO.
6	R	1	TSR_EMPTY	<p>Both Transmitter Shift Register (TSR) and TX FIFO are empty indicator</p> <ul style="list-style-type: none"> ● 0:Otherwise ● 1: This bit is set to logic 1 whenever the Transmitter FIFO (TX FIFO) and the Transmitter Shift Register (TSR) are both empty. It is reset to logic 0 whenever either the TX FIFO or TSR contains a data character.
5	R	1	TXFIFO_EMPTY	<p>TXFIFO empty indicator.</p> <ul style="list-style-type: none"> ● 0: Otherwise ● 1: It indicates that the Transmitter FIFO is empty. This bit is set when the Transmitter FIFO is empty; it is cleared when at least 1 byte is written to the Transmitter FIFO.
4	R	0	BREAK_ERR_INT	<p>Break Interrupt (BI) indicator</p> <ul style="list-style-type: none"> ● 0: No break condition in the current character ● 1: Set 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 (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status Register. This

				error is revealed to the CPU when its associated character is at the top of the FIFO. When break occurs only one zero character is loaded into the FIFO. The next character transfer is enabled after Serial-In goes to the marking state (Logic 1) and receives the next valid start bit.
3	R	0	FRAMING_ERR	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 did not have a valid stop bit. Of course, generally, it might be that all the following data is corrupt. It indicates that the received character did not have a valid stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a logic 0 bit (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status Register. This error is revealed to the CPU when its associated character is at the top of the FIFO. The UART will try to resynchronize after a framing error. To do this it assumes that the framing error was due to the next start bit, so it samples this "start" bit twice and then takes in the "data".
2	R	0	PARITY_ERR	Parity Error (PE) indicator <ul style="list-style-type: none"> ● 0: No parity error in current character ● 1: Indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to logic "1" upon detection of a parity error and is reset to logic 0 whenever the CPU reads the contents of the Line Status Register. This error is revealed to the CPU when its associated character is at the top of the FIFO (next character to be read).
1	R	0	OVERRUN_ERR	Overrun Error (OE) indicator <ul style="list-style-type: none"> ● 0: No Overrun state ● 1: Indicates that data in the RX FIFO is not read by the CPU before the next character was transferred into the RX FIFO, thereby destroying the previous character. The OE indicator is set to logic 1 upon detection of an overrun condition and reset whenever the CPU reads the contents of the Line Status Register. If the FIFO mode data continues to fill the FIFO beyond the trigger level, an overrun error will occur only after the FIFO is full and the next character has been completely received in the shift register. OE is indicated to the CPU as soon as it happens. The character in the shift register is overwritten, but it is not transferred to the FIFO.
0	R	0	RXFIFO_DATARDY	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. <p>Bit0 is reset to logic "0" by reading all of the data in the Receiver Buffer Register or the RX FIFO.</p>

10.3.9 MSR

- Name: Modem Status Register
- Width: 32bit
- Initial Value: 0x0000
- Note: The register displays the current state of the modem control lines to the CPU. Also, four bits also provide an indication in the state of one of the modem status lines. These bits are set to “1” when a change in corresponding line has been detected and they are reset when the register is being read.

Whenever bit 0, 1, 2, 3 is set to logic 1, a MODEM Status Interrupt is generated.

REG 10-12 (Offset 0018h) MSR

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7	R	0	R_DCD	Complement of the DCD input or equals to OUT2(MCR[3]) in loopback mode.
6	R	0	R_RI	Complement of the RI input or equals to OUT1(MCR[2]) in loopback mode.
5	R	0	R_DSR	Complement of the DSR input or equals to DTR(MCR[0]) in loopback mode.
4	R	0	R_CTS	Complement of the CTS input or equals to RTS(MCR[1]) in loopback mode.
3	R	0	D_DCD	Delta Data Carrier Detect (DDCD) indicator <ul style="list-style-type: none"> ● 0: Otherwise ● 1: The DCD line has changed its state
2	R	0	TERI	Trailing Edge of Ring Indicator (TERI) detector. <ul style="list-style-type: none"> ● 0: Otherwise ● The RI line has changed its state from low to high state
1	R	0	D_DSR	Delta Data Set Ready (DDSR) indicator <ul style="list-style-type: none"> ● 0: Otherwise ● 1: The DSR line has changed its state
0	R	0	D_CTS	Delta Clear to Send (DCTS) indicator <ul style="list-style-type: none"> ● 0: Otherwise ● 1: The CTS line has changed its state

10.3.10 SCR

- Name: Scratch Pad Register
- Width: 32bit
- Initial Value: 0x00000030
- Note: The 32-bit Read/Write register is for additional control function.

$$\text{AverageBaudRate} = \frac{\text{clock_freq}}{(ovsr[3:0]+5 + \frac{ovsr_adj[0]+ovsr_adj[1]+\dots+ovsr_adj[10]}{11})*(\text{divisor}[15:0])}$$

REG 10-13 (Offset 001Ch) SCR

Bit	Access	INI	Symbol	Description
31:27	R	0	RSVD	
26:16	R/W	0	XFACTOR_ADJ[10:0]	One factor of Baud rate calculation, i.e., the ovsr_adj[10:0] of AverageBaudRate formula.
15:13	R	0	RSVD	
12	R	0	RSVD	
11:8	R/W	0	DBG_SEL[3:0]	Debug port selection
7	R/W	0	RX_BREAK_SIGNAL_INTERRUPT_STATUS	Rx break signal interrupt status, Write 1 to this bit, it will clear the interrupt pending status.
6	R/W	0	RX_BREAK_SIGNAL_INTERRUPT_ENABLE	Rx break signal interrupt enable
5	R/W	1	FL_SET_BI_ERR	Set BI error flag
4	R/W	1	FL_FRAME_ERR	Frame error flag
3	R/W	0	PIN_LB_TEST	For UART IP txd/rxd/rts/cts pin loopback test
2:0	R/W	0	RSVD	

10.3.11 STSR

- Name: STS Register
- Width: 32bit
- Initial Value: 0x0000
- Note: The 32-bit Read/Write register is for additional control function.

$$\text{AverageBaudRate} = \frac{\text{clock_freq}}{(ovsr[3:0]+5 + \frac{ovsr_adj[0]+ovsr_adj[1]+\dots+ovsr_adj[10]}{11})*(\text{divisor}[15:0])}$$

REG 10-14 (Offset 0028h) STSR

Bit	Access	INI	Symbol	Description
31:29	R	0	RSVD	
28:27	R	2'b11	RXFIFO_TRIGGER_LEVEL	rxfifo_trigger_level in FCR bit[7:6], Define the 32-entries Receiver FIFO Interrupt trigger level 0~31 bytes <ul style="list-style-type: none"> ● 00: 1 byte ● 01: 8 bytes ● 10: 16 bytes ● 11: 28 bytes (for some device detect RTS de-assertion slower, reserve more RX FIFO space to prevent RX FIFO overflow)
26	R	0	TXFIFO_LOW_LEVEL	txfifo_low_level in FCR bit[5], Define the Transmission FIFO Low Water Level Interrupt trigger. <ul style="list-style-type: none"> ● 0: 4 byte

				● 1: 8 bytes
25	R	1	EN_RXFIFO_ERR	En_rx fifo_err in FCR bit[0], Set as 1 to enable the report of Error in RCVR FIFO field in LSR bit [7].
24	R	1	DMA_MODE	dma_mode field of FCR bit[3]
23:8	R	0	RSVD	
7:4	R/W	4'hB	XFACTOR	Factor of Baud rate calculation, i.e., the ovsr[3:0] of AverageBaudRate formula.
3	R/W	0	RESET_RCV	Reset UART Receiver
2:0	R	0	RSVD	

10.3.12 RBR

- Name: Receiver Buffer Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register read the received byte from RX FIFO. The register is a read-only register

REG 10-15 (Offset 0024h) RBR

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7:0	R	0	RXDATA	Rx Data ● Note: Bit 0 is the least significant bit. It is the first bit serially received.

10.3.13 THR

- Name: Transmitter Holding Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to write the transmitted byte to TX FIFO. The register is a write-only register.

REG 10-16 (Offset 0024h) THR

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7:0	W	0	TXDATA	Tx Data ● Note: Bit 0 is the least significant bit. It is the first bit serially transmitted.

10.3.14 MISCCR

- Name: MISC Control Register
- Width: 32bit
- Initial Value: 0x00000426
- Note: This register is for the control of Irda SIR mode relative, and TXDMA / RXDMA relative control.

REG 10-17 (Offset 0028h) MISCCR

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	IRDA_RX_INV	<ul style="list-style-type: none"> ● 0: Don't invert irda_rx_i (It is default) ● 1: Invert irda_rx_i ● Note: Default IRDA RX pulse is Low Active
14	R/W	0	IRDA_TX_INV	<ul style="list-style-type: none"> ● 0: Don't invert irda_tx_o (It is default) ● 1: Invert irda_tx_o ● Note: Default IRDA TX Pulse is High Active
13:8	R/W	6'h4	RXDMA_BURSTSIZE[5:0]	Rx DMA burst size
7:3	R/W	5'h4	TXDMA_BURSTSIZE[4:0]	Tx DMA burst size
2	R/W	1	RXDMA_EN	<ul style="list-style-type: none"> ● 0: Rx DMA is disabled ● 1: Rx DMA is enabled (valid when DMA_MODE in FCR[3] is 1'b1)
1	R/W	1	TXDMA_EN	<ul style="list-style-type: none"> ● 0: Tx DMA is disabled ● 1: Tx DMA is enabled (valid when DMA_MODE in FCR[3] is 1'b1)
0	R/W	0	IRDA_ENABLE	<ul style="list-style-type: none"> ● 0: UART mode only ● 1: UART co-work with IRDA SIR mode. i.e., txd/rxd are irda signals.

10.3.15 IRDA_TXPLSR

- Name: IRDA SIR TX Pulse Width Control Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is for the control of TX pulse width and relative position, only valid when irda_enable is 1'b1.
 - ◆ Specification description of Signaling Rate and Pulse Duration

An IrDA serial infrared interface must operate at 9.6 kb/second. Additional allowable rates listed below are optional. Signaling rate and pulse duration specification are shown in following table.

For all signaling rates up to and including 115.2kb/s the minimum pulse duration is the same (the specification allows both a 3/16 of bit duration pulse and a minimum pulse

duration for the 115.2kb/s signal (1.63us minus the 0.22us tolerance)). The maximum pulse duration is 3/16 of the bit time, pulse the greater of the tolerance of 2.5% of the bit duration, or 0.6us.

Table 10-18 Signaling Rate and Pulse Duration Specification

Signaling Rate	Modulation	Rate Tolerance % of Rate	Pulse Duration Minimum	Pulse Duration Nominal	Pulse Duration Maximum
2.4kb/s	RZI	+/- 0.87	1.41 us	78.13 us	88.55 us
9.6kb/s	RZI	+/- 0.87	1.41 us	19.53 us	22.13 us
19.2kb/s	RZI	+/- 0.87	1.41 us	9.77 us	11.07 us
38.4kb/s	RZI	+/- 0.87	1.41 us	4.88 us	5.96 us
57.6kb/s	RZI	+/- 0.87	1.41 us	3.26 us	4.34 us
115.2kb/s	RZI	+/- 0.87	1.41 us	1.63 us	2.23 us

REG 10-19 (Offset 002Ch) IRDA_TXPLSR

Bit	Access	INI	Symbol	Description
31	R/W	0	UPPERBOUND_SHIFTRIGHT	<ul style="list-style-type: none"> ● 0: shift left (minus offset value of txplsr[30:16]) ● 1: shift right (plus offset value txplsr[30:16])
30:16	R/W	0	TXPULSE_UPPERBOUND_SHIFTVAL	<p>The shift value of SIR Tx pulse's right edge position. The nominal IRDA SIR Tx pulse width is 3/16 bit time. If user wants to increase or decrease the right edge position of Tx pulse. Must change the value.</p> <ul style="list-style-type: none"> ● Note: In time unit of uart clock period. Nominally the 10ns.(100Mhz uart clk)
15	R/W	0	LOWBOUND_SHIFTRIGHT	<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	R/W	0	TXPULSE_LOWBOUND_SHIFTVAL	<p>The shift value of SIR Tx pulse's left edge position. The nominal IRDA SIR Tx pulse width is 3/16 bit time. If user wants to increase or decrease the left edge position of Tx pulse. Must change the value.</p> <ul style="list-style-type: none"> ● Note: In time unit of uart clock period. Nominally the 10ns.(100Mhz uart clk)

10.3.16 DBG_UART

- Name: Debug UART Register
- Width: 32bit
- Initial Value: 0x0000
- Note: The 32-bit register is for the debug purpose, which identifies the status of the internal UART status. It is a read-only register.

REG 10-20 (Offset 003Ch) DBG_UART

Bit	Access	INI	Symbol	Description
31:0	R	0	DBG_UART	Debug port output value. It depends on dbg_sel value in SCR[2:0].

10.3.17 RFCR

- Name: RX Filter Control Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This 32-bit register is for the RX filter configuration.

REG 10-21 (Offset 0040h) RFCR

Bit	Access	INI	Symbol	Description																	
31:8	R	0	RSVD																		
7	R/W	0	RF_EN	<p>RX Filter enable control</p> <ul style="list-style-type: none"> ● 0: Disable ● 1: Enable ● Note: When RF_EN set as 1, the other relevant parameter (registers in RX Filter Control Register/ RX Filter Magic Pattern Register/ RX Filter Mask Value Register/ RX Filter Timeout Register) must be set ready. 																	
6:4	R	0	RSVD																		
3:2	R/W	0	RF_CMP_OP	<p>Set the RX filter comparing rule. The “matched” condition is listed as following table:</p> <table border="1"> <thead> <tr> <th>RF_LEN</th> <th>RF_CMP_OP</th> <th>Matched Condition</th> </tr> </thead> <tbody> <tr> <td rowspan="3">0</td> <td>0(AND)</td> <td>1st byte = Magic Pattern1</td> </tr> <tr> <td>1(OR)</td> <td>(1st byte = Magic Pattern1) OR (1st byte = Magic Pattern2)</td> </tr> <tr> <td>2(XOR)</td> <td>(1st byte != Magic Pattern1) AND (1st byte != Magic Pattern2)</td> </tr> <tr> <td rowspan="3">1</td> <td>0(AND)</td> <td>(1st byte = Magic Pattern1) AND (2nd byte = Magic Pattern2)</td> </tr> <tr> <td>1(OR)</td> <td>(2nd byte = Magic Pattern1) OR (2nd byte = Magic Pattern2)</td> </tr> <tr> <td>2(XOR)</td> <td>(1st byte != Magic Pattern1) AND (2nd byte != Magic Pattern2)</td> </tr> </tbody> </table>	RF_LEN	RF_CMP_OP	Matched Condition	0	0(AND)	1 st byte = Magic Pattern1	1(OR)	(1 st byte = Magic Pattern1) OR (1 st byte = Magic Pattern2)	2(XOR)	(1 st byte != Magic Pattern1) AND (1 st byte != Magic Pattern2)	1	0(AND)	(1 st byte = Magic Pattern1) AND (2 nd byte = Magic Pattern2)	1(OR)	(2 nd byte = Magic Pattern1) OR (2 nd byte = Magic Pattern2)	2(XOR)	(1 st byte != Magic Pattern1) AND (2 nd byte != Magic Pattern2)
RF_LEN	RF_CMP_OP	Matched Condition																			
0	0(AND)	1 st byte = Magic Pattern1																			
	1(OR)	(1 st byte = Magic Pattern1) OR (1 st byte = Magic Pattern2)																			
	2(XOR)	(1 st byte != Magic Pattern1) AND (1 st byte != Magic Pattern2)																			
1	0(AND)	(1 st byte = Magic Pattern1) AND (2 nd byte = Magic Pattern2)																			
	1(OR)	(2 nd byte = Magic Pattern1) OR (2 nd byte = Magic Pattern2)																			
	2(XOR)	(1 st byte != Magic Pattern1) AND (2 nd byte != Magic Pattern2)																			
1	R/W	0	RF_MASK_EN	<p>Enable the mask operation for received data.</p> <ul style="list-style-type: none"> ● 0: No mask, compare the received bytes with the magic pattern directly. ● 1: Mask enabled, received data needs to “AND” with mask value before it to be compared with magic pattern. 																	
0	R/W	0	RF_LEN	<p>Set the length of received data to be check.</p> <ul style="list-style-type: none"> ● 0: Check the first 1 received byte only. ● 1: Check the first 2 received bytes. 																	

10.3.18 RFMPR

- Name: RX Filter Magic Pattern Register
- Width: 32bit
- Initial Value: 0x0000FFFF

- Note: This 32-bit register is for storing the magic pattern for the RX filter checking.

REG 10-22 (Offset 0044h) RFMPR

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:8	R/W	8'hFF	RF_MP2	The magic pattern2 for the 2 nd received byte checking.
7:0	R/W	8'hFF	RF_MP1	The magic pattern1 for the 1 st received byte checking.

10.3.19 RFMVR

- Name: RX Filter Mask Value Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This 32-bit register is for storing the mask value for RX bytes. If the mask is enabled, the first 1 or 2 received bytes needs to do a “AND” operation with this mask value before it be compared with magic pattern.

REG 10-23 (Offset 0048h) RFMVR

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:8	R/W	0	RF_MV2	The mask value for the 2 nd received byte.
7:0	R/W	0	RF_MV1	The mask value for the 1 st received byte.

The rule of the RX Filter checking is listed as the table below:

Table 10-24 RX Filter Matching Condition

RF_LEN	RF_CMP_OP	RF_MASK_EN	Matched Condition
0	0 (AND)	0	(1 st byte = RF_MP1)
		1	((1 st byte & RF_MV1) = RF_MP1)
	1 (OR)	0	(1 st byte = RF_MP1) OR (1 st byte = RF_MP2)
		1	((1 st byte & RF_MV1) = RF_MP1) OR ((1 st byte & RF_MV2) = RF_MP2)
	2 (XOR)	0	(1 st byte != RF_MP1) AND (1 st byte != RF_MP2)
		1	((1 st byte & RF_MV1) != RF_MP1) AND ((1 st byte & RF_MV2) != RF_MP2)
	1	0	(1 st byte = RF_MP1) AND (2 nd byte = RF_MP2)
		1	((1 st byte & RF_MV1) = RF_MP1) AND ((2 nd byte & RF_MV2) = RF_MP2)
		0	(1 st byte = RF_MP1) OR (2 nd byte = RF_MP2)
		1	((1 st byte & RF_MV1) = RF_MP1) OR ((2 nd byte & RF_MV2) = RF_MP2)
2	0 (AND)	0	(1 st byte != RF_MP1) AND (2 nd byte != RF_MP2)
		1	((1 st byte & RF_MV1) != RF_MP1) AND ((2 nd byte & RF_MV2) != RF_MP2)

10.3.20 RFTOR

- Name: RX Filter Timeout Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This 32-bit register is for the timeout setting of the RX idle state detection. This timer is a down counter with tick time equal to a UART bit time which is from the baud rate generator. This timer is used to detect the idle state of the UART receiving. The RX idle state means there is no data received for a certain time. This register is used to set the timeout by assign a number of tick time. When the 1st byte is received, the timer will load the initial value from this register and then start to countdown. The timer will reload the initial value whenever a new byte is received. The timeout occurred when this timer count to 0, the RX filter will be reset as initialed state.

REG 10-25 (Offset 004Ch) RFTOR

Bit	Access	INI	Symbol	Description
31:20	R	0	RSVD	
19:0	R/W	0	RF_TIMEOUT	Set the timeout value of the RX filter idle detection. This value is number of ticks. A tick time is equal to a UART bit time.

10.3.21 RFLVR

- Name: RX FIFO Level Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is for the SW can read the number of bytes in the RX FIFO.

REG 10-26 (Offset 0050h) RFLVR

Bit	Access	INI	Symbol	Description
31:6	R	0	RSVD	
5:0	R	0	RX_FIFO_LV	The level of the RX FIFO. This value indicates the number of data bytes in the RX FIFO.

10.3.22 TFLVR

- Name: TX FIFO Level Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is for the SW can read the number of bytes in the TX FIFO.

REG 10-27 (Offset 0054h) TFLVR

Bit	Access	INI	Symbol	Description
31:5	R	0	RSVD	
4:0	R	0	TX_FIFO_LV	The level of the TX FIFO. This value indicates the number of data bytes in the TX FIFO.

10.3.23 VDR_INTSR

- Name: Vendor Interrupt Status Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to read the Vendor interrupt status.

REG 10-28 (Offset 0058h) VDR_INTSR

Bit	Access	INI	Symbol	Description
31:4	R	0	RSVD	
3	R/W	0	RXIDLE_TIME_OUT_INT_STS	Interrupt status of RX IDLE timeout. Write 1 to this bit will clear the pending status.
2	R/W	0	TXFIFO_LOW_STS	This bit indicates the interrupt pending status of the TX FIFO water level equal to the level setting. The software can use this interrupt to know the TX FIFO water level is low and refill the TX FIFO. To enable this interrupt, IER bit[5] should be written as 1. Write 1 to this bit will clear the pending status.
1	R/W	0	RXFILTER_IDLE_TIMEOUT_STS	The RX idle detection mechanism is used to check whether the receiving is in idle state for re-acquire pattern matching. This mechanism uses a timer to monitor new data receiving. This timer will reload the initial value whenever a new byte is received. If the timeout occurred (timer value countdown to 0), the RX filter will be reset to the initialized state and restart the first received 1 or 2 bytes checking.
0	R/W	0	RXFILTER_PATTERNMATCH_STS	The interrupt status of RX filter pattern checking matched. Write 1 to clear this interrupt status. <ul style="list-style-type: none"> ● 0: Interrupt is not pending. ● 1: Interrupt is pending.

10.3.24 VDR_INTER

- Name: Vendor Interrupt Enable Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to enable the Vendor interrupt.

REG 10-29 (Offset 005Ch) VDR_INTER

Bit	Access	INI	Symbol	Description
31:4	R	0	RSVD	
3	R/W	0	RXIDLE_TIMEOUT_INT_EN	The RX idle timeout interrupt enabling control: ● 0: disable ● 1: enable
2	R/W	0	TXFIFO_LOW_INT_EN	The TX FIFO water level interrupt enabling control: ● 0: disable ● 1: enable
1	R/W	0	RXFILTER_IDLE_TIMEOUT_INT_EN	The Rx filter idle timeout interrupt enabling control: ● 0: disable ● 1: enable
0	R/W	0	RXFILTER_PATTERNMATCH_INT_EN	The Rx filter pattern matched interrupt enabling control: ● 0: disable ● 1: enable

10.3.25 RXIDLE_TOCR

- Name: RX Idle Timeout Control Register
- Width: 32bit
- Initial Value: 0x0000
- Note: The 32-bit register controls the RX transceiver timeout enable, and rx_idle timeout character times. When RX UART is idle and rxfifo is empty for a several dedicated character times. The RX timeout will assert an interrupt to notify CPU

REG 10-30 (Offset 0060h) RXIDLE_TOCR

Bit	Access	INI	Symbol	Description
31	R/W	0	RXIDLE_TIMEOUT_EN	RX_IDLE_TIMEOUT Enable, default 0. 0: disable 1: enable
30:4	R	0	RSVD	
3:0	R/W		RXIDLE_TIMEOUT_VAL[3:0]	Default 0. 4'd0: 8 bit time. (1*8) 4'd1: 16 bit time. (2*8) 4'd2: 32 bit time. (2^2*8) 4'd3: 64 bit time. (2^3*8) 4'd4: 128 bit time. (2^4*8)

				4'd5: 256 bit time. (2^{5*8}) 4'd6: 512 bit time. (2^{6*8}) 4'd7: 1024 bit time. (2^{7*8}) 4'd8: 2048 bit time. (2^{8*8}) 4'd9: 4096 bit time. (2^{9*8}) 4'd10: 8192 bit time. (2^{10*8}) 4'd11: 16384 bit time. (2^{11*8}) 4'd12: 32768 bit time. (2^{12*8}) 4'd13: 65535 bit time. (2^{13*8}) 4'd14: 131072 bit time. (2^{14*8}) 4'd15: 262144 bit time. (2^{15*8})
--	--	--	--	---

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

11 Serial Peripheral Interface (SPI)

11.1 Overview

11.1.1 Application Scenario

Serial Peripheral Interface (SPI) is a serial interface that enables data communication between microcontrollers and other peripherals. High throughput and the full-duplex capability with simple hardware interface makes SPI very efficient for various application. SPI is usually used to talk to variety of peripherals including sensors, control devices, memory, LCD, SD card etc.

11.1.2 Features

- Follow Advanced Peripheral Bus (APB) 2 protocol specification
- Support three interfaces
 - Motorola Serial Peripheral Interface (SPI)
 - Texas Instruments Serial Protocol (SSP)
 - National Semiconductor Microwire
- One SPI device, it can be either a master or a slave
- Maximum speed support for each SPI interface is listed below:

Table 11-1 SPI Speed

	Master	Slave
SPI 0	20 MHz	5 MHz (Receive only) 4 MHz (Transmit/Receive)

The speed is measured under DMA mode which data stream is handling by DMA handshake interfaces. By using DMA handshake interfaces, data stream is transferred more efficiently because CPU does not have to interrupt the transfer to handle data stream.

- Support DMA handshaking interface to enable DMA transfer with SPI

To ensure the SPI device is able to operate at high speed, we suggest users to adopt DMA mode instead of Interrupt mode for data stream transfer. The maximum data rate of interrupt mode transfer is easily affected by the loading of the processor. If the processor cannot handle the interrupt in time, SPI FIFO may be overflow or underflow. SPI transfer with DMA mode is initiated via API functions with a suffix “dma”.

- Support 8 bit and 16 bit data frame size

- Programmable clock polarity (SCPOL) and clock phase (SCPH) for SPI interface

Table 11-2 SPI Clock Polarity

	SCPOL = 0	SCPOL = 1
SCPH = 0	Mode 0	Mode 2
SCPH = 1	Mode 1	Mode 3

- Support bit swapping and byte swapping features
- The depth of transmit FIFO and receive FIFO are 1024 bits
 - 64 data frames at most

11.1.3 Architecture

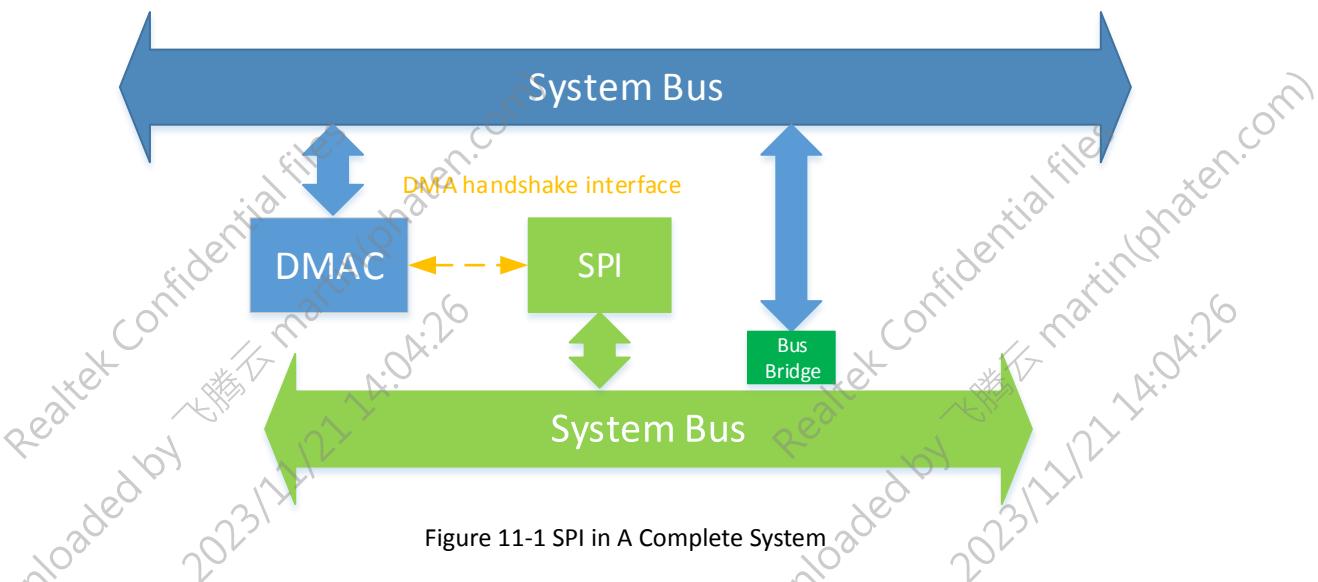


Figure 11-1 SPI in A Complete System

The Ameba-Z II SPI is an APB bus slave device. The data transfer and register access are performed on the APB bus. An APB Bridge, which becomes an APB master, manages all bus activities and communicates with AXI bus. Processor and DMAC on the AXI bus access SPI through this APB Bridge. They rely on the bridge to transform AXI bus signals to APB signals recognizable by SPI APB interface. A DMA handshake interface is introduced between DMAC and SPI to allow SPI to transfer data by DMAC without intervention of processor.

The clock source of the APB bus is pclk. All control units of the SPI except FIFO are operating on the pclk domain. Interrupt control and register access are all based on pclk cycles to sample signals. The other clock domain sclk is generated by Clock Pre-scale unit. This sclk is the actual operating frequency which the SPI device communicates with other peripherals.

11.1.3.1 Functional Block Diagram

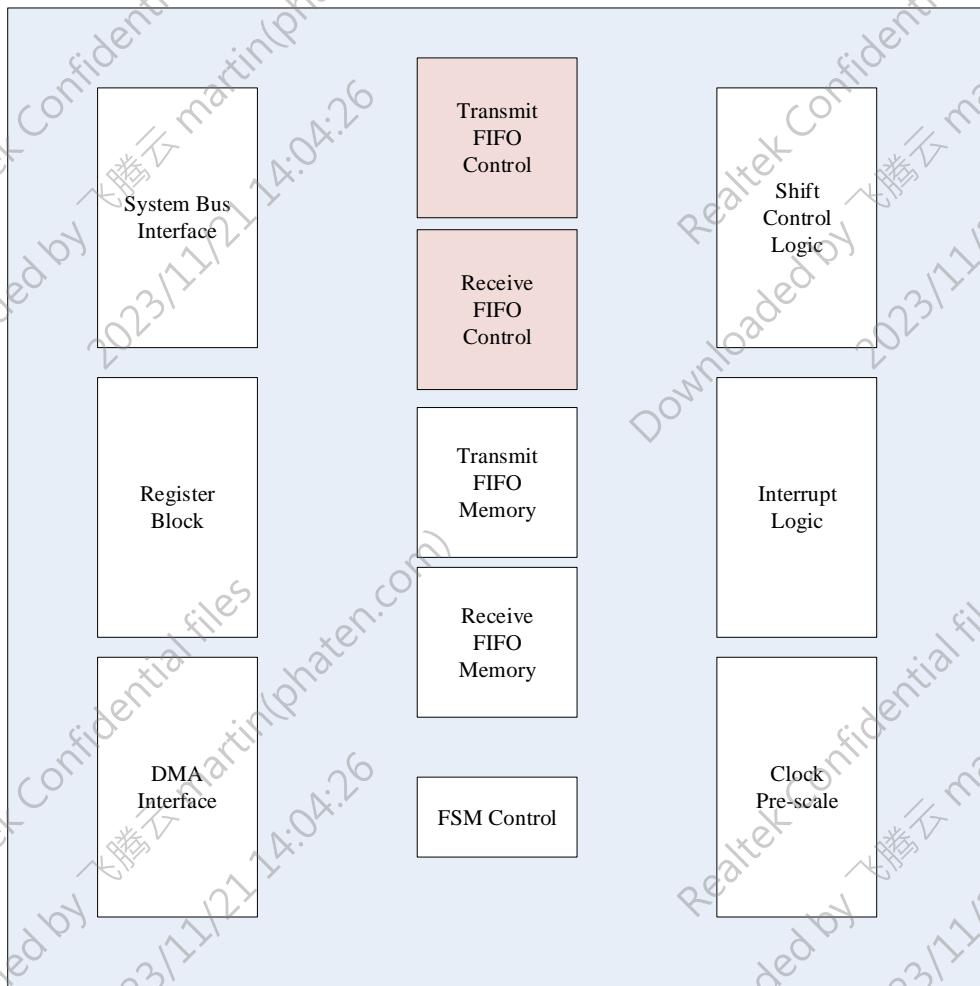


Figure 11-2 SPI Block Diagram

- APB interface: A bus interface for low bandwidth control accesses. The processor accesses data, control, and status information on the SPI device through this APB interface.
- Register Block: Contains configuration registers and is the interface with software.
- DMA interface: Generates the handshaking signals to the central DMA controller in order to automate the data transfer without CPU intervention. The hardware handshake interface plays an important role in controlling data flow so that data overflow and underflow can be avoided.
- Transmit FIFO Control / Receive FIFO Control: Record the level of the FIFO.
- Transmit FIFO Memory / Receive FIFO Memory: Store data in the FIFO or remove data out of FIFO. The depth of each FIFO is 64 bytes.
- FSM Control: Finite State Machine Control controls the state change of the SPI device.
- Shift Control Logic: Shift out data out of FIFO bit by bit.
- Interrupt Logic: Generates the raw interrupt and interrupt flags, allowing them to be set

and to be cleared.

- Clock Pre-scale: Generate sclk from pclk. The sclk is configurable by setting BAUDR register value.

11.2 Functional Description

11.2.1 Overview

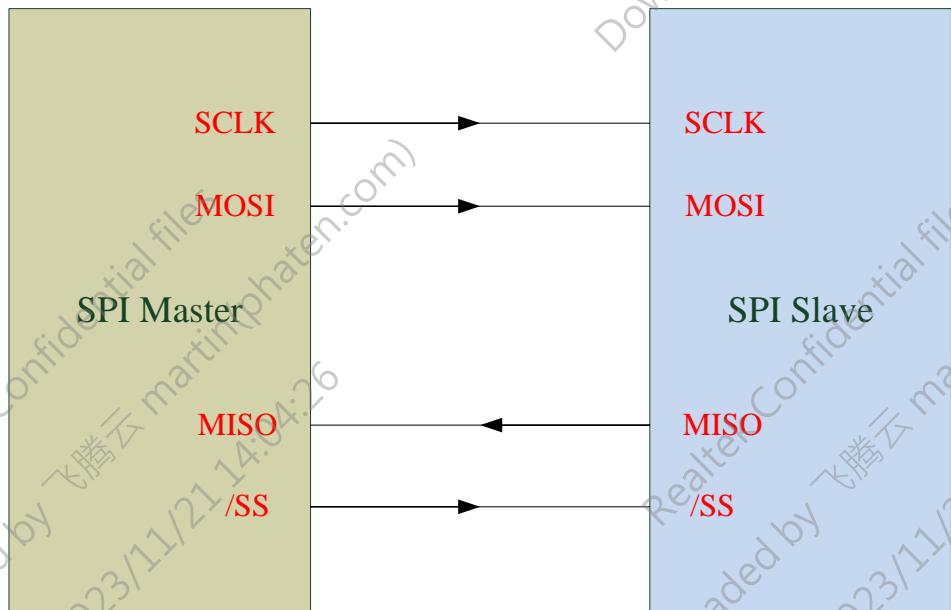


Figure 11-3 SPI Pin Connection

The SPI device is a full-duplex master or slave-synchronous serial interface. The host processor accesses data, control, and status information on the SPI device through the APB interface. The SPI may also interface with a DMA Controller using an optional set of DMA signals, which can be selected at configuration time.

The SPI can be configured in one of two modes of operations: as a serial master or a serial slave. When the device is configured as a master device, it generates SCLK to the slave device to transfer data bit by bit. The SPI can connect to any serial-master or serial-slave peripheral device using one of the following interfaces:

- Motorola Serial Peripheral Interface (SPI)
- Texas Instruments Serial Protocol (SSP)
- National Semiconductor Microwire

The SPI transfer is accomplished by four signals. SCLK is a clock signal provided by SPI master, data is latched on the rising or falling edge of the clock. MOSI indicates master output slave input signal while MISO means master input slave output signal. Data are transferred through these two signals. /SS is the slave select line, or chip select line, which is controlled by SPI master. SPI master determines which slave device to communicate with by selecting the corresponding slave select signal. Only one slave selection is supported. Once /SS is selected and data is pushed into SPI master's transmit FIFO, the clocks start to toggle to start a transfer.

Data size of one data item transferred by the SPI is configurable. Ameba-Z II SPI supports 8 bits per data frame or 16 bits per data frame to satisfy different application. For data streaming transfer, users can further control whether to toggle /SS signal between data frames that makes SPI compatible with various peripherals.

11.2.2 SPI Transfer Protocol

11.2.2.1 Motorola Serial Peripheral Interface (SPI)

SPI interface has two control factors, SCPH and SCPOP, to determine data latch timing. The clock polarity (SCPOP) configuration parameter determines whether the inactive state of the serial clock is high or low.

The clock phase (SCPH) decides data is captured on the rising edge or the falling edge of the slave select signal (/SS). When 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 MOSI & MISO prior to the first serial clock edge. When SCPH = 1, both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is active. The first data bit is captured on the second serial clock edge, i.e. trailing edge. Data are propagated by the master and slave peripherals on the rising edge of the serial clock.

To ensure data can be transmitted and received correctly, both SPI peripherals must have identical serial clock phase and clock polarity settings.

- SCPOP = 0 (Inactive state of serial clock is low)
SCPH = 0 (Serial clock toggles in middle of first data bit)

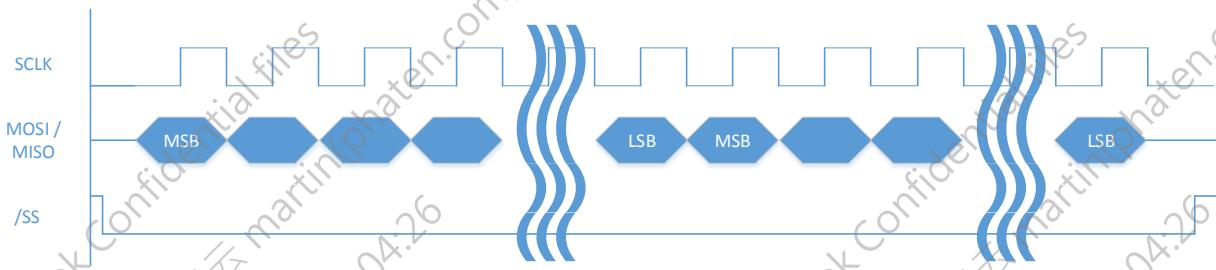


Figure 11-4 SPI Protocol: Mode 0

- SCPOL = 0 (Inactive state of serial clock is low)
- SCPH = 1 (Serial clock toggles at start of first data bit)

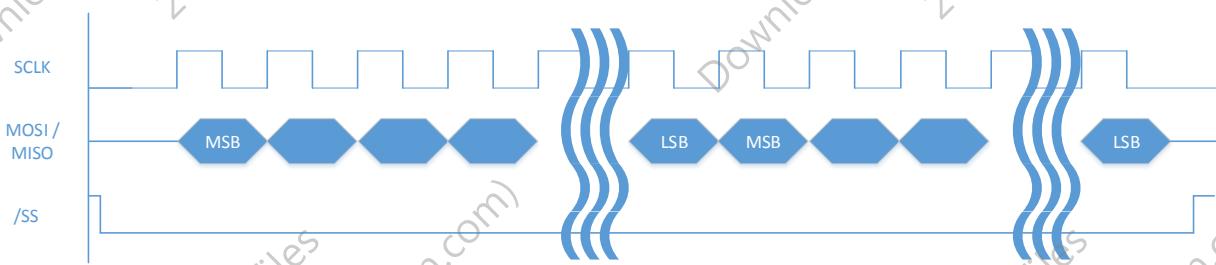


Figure 11-5 SPI Mode Protocol: Mode 1

- SCPOL = 1 (Inactive state of serial clock is high)
- SCPH = 0 (Serial clock toggles in middle of first data bit)

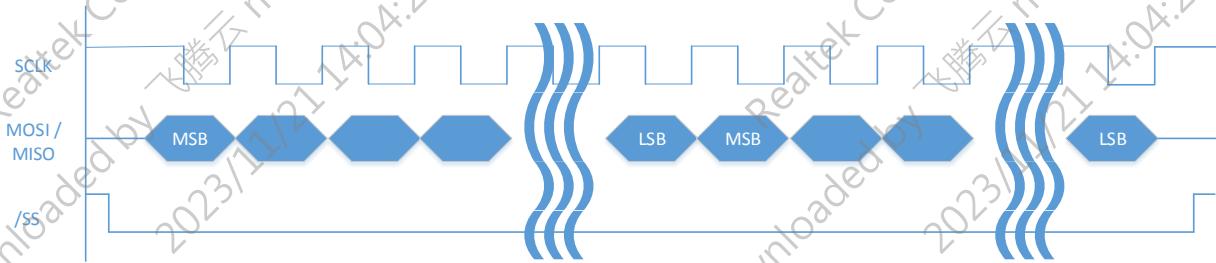


Figure 11-6 SPI Mode Protocol: Mode 2

- SCPOL = 1 (Inactive state of serial clock is high)
- SCPH = 1 (Serial clock toggles at start of first data bit)

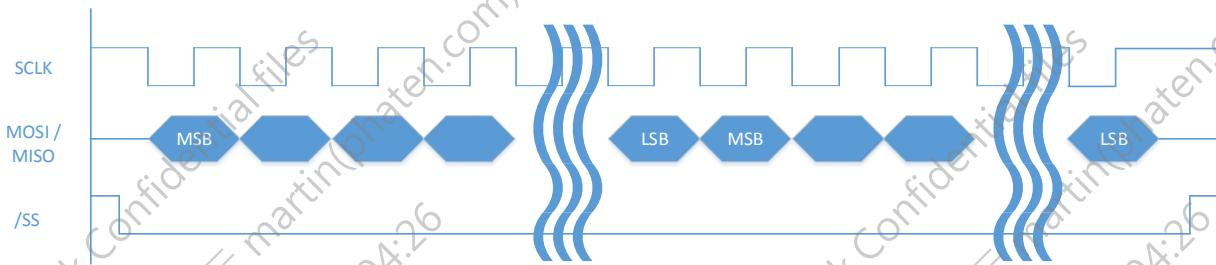


Figure 11-7 SPI Mode Protocol: Mode 3

11.2.2.2 Texas Instruments Synchronous Serial Protocol (SSI)

Data transfers begin by asserting frame indicator line (/SS) for one serial clock period. Data are propagated on the rising edge of the serial clock and captured on the falling edge. When continuous data frames are transferred, the frame indicator is asserted for one clock period during the same cycle as the LSB from the current transfer, indicating that another data frame follows.

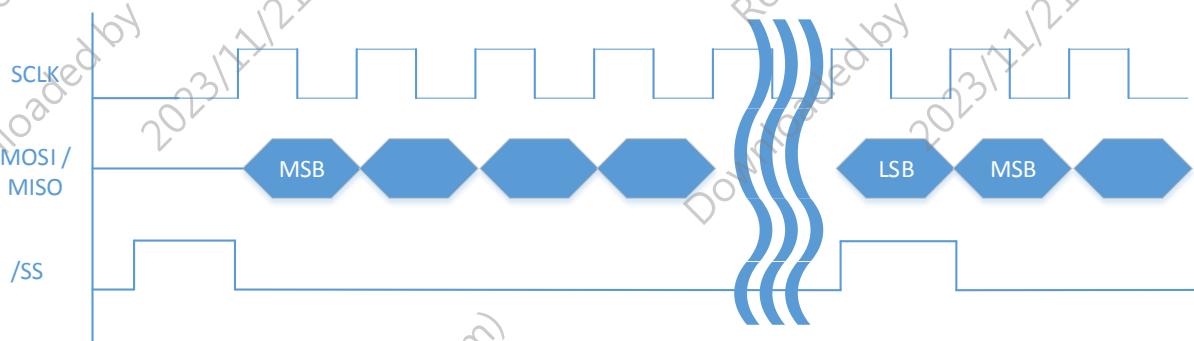


Figure 11-8 SSI Protocol

11.2.3 SPI Interrupts

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.

11.2.3.1 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.

11.2.3.2 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).

11.2.3.3 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.

11.2.3.4 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).

11.2.3.5 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).

11.2.3.6 Multi-Master Contention Interrupt (ssi_mst_intr)

Available only when the SPI is a serial master. 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).

11.2.3.7 Transmit FIFO Underflow Interrupt (ssi_txu_intr)

Available only when the SPI is a serial slave. When the FIFO is empty but SPI master sends clock to the slave device, this interrupt is triggered. This interrupt bit can be cleared by reading transmit FIFO underflow interrupt clear register (TXUICR).

11.2.3.8 SS_N Rising Edge Detect Interrupt (ssi_icr_intr)

Available only when the SPI is a serial slave. When /SS returns from low to high, SPI becomes inactive state. This rising edge of /SS is detected by the interrupt. Read SS_N rising edge detect interrupt clear register (SSRICR) clears this interrupt state.

11.2.3.9 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 interrupt requests. Read interrupt clear register (ICR) can clear all interrupt pending bits.

11.3 Control Registers

11.3.1 SPI_CTRLR0

- **Name:** SPI Control Register 0
- **Size:** 16 bits
- **Initial Value:** 0x0

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.

REG 11-3 (Offset 0000h) SPI_CTRLR0

Bits	Access	INI	Symbol	Description
31	R/W	0x0	SS_T	SPI master sets this bit to determine whether to toggle /SS between successive frames or not. SPI slave sets this bit depending on the /SS behavior of the master. SPI device should be aware of the /SS behavior of the other SPI device communicated with it and set the correct value accordingly. 0 – /SS does not toggle between successive frames. 1 – /SS toggles between successive frames.
24	R/W	0x0	RX_BIT_SWAP	Swap order of receive bit
23	R/W	0x0	RX_BYTE_SWAP	Swap order of receive byte
22	R/W	0x0	TX_BIT_SWAP	Swap order of transmit bit
21	R/W	0x0	TX_BYTE_SWAP	Swap order of transmit byte
15:12	R/W	0x0	CFS	Control Frame Size. Select the length of the control word for the Microwire frame format.
10	R/W	0x0	SLV_OE	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	R/W	0x0	TMOD	<p>Transfer Mode. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or the transmit data are valid.</p> <p>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.</p> <p>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.</p> <p>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. SPI 3 only supports transmit & receive mode.</p> <p>00 – Transmit & Receive 01 – Transmit Only 10 – Receive Only</p>
7	R/W	0x0	SCPOL	<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	R/W	0x0	SCPH	<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 first data bit 1: Serial clock toggles at start of first data bit</p>
5:4	R/W	0x0	FRF	<p>Frame Format. Selects which serial protocol transfers the data.</p> <p>00 – Motorola SPI 01 – Texas Instruments SSP 10 – National Semiconductors Microwire 11 – Reserved</p>
3:0	R/W	0x0	DFS	<p>Data Frame Size. Selects the data frame length.</p> <p>0111 – 8-bit serial data transfer 1111 – 16-bit serial data transfer</p>

11.3.2 SPI_CTRLR1

- **Name:** SPI Control Register 1
- **Size:** 16 bits
- **Initial Value:** 0x0

This register exists only when the SPI is configured as a master device.

REG 11-4 (Offset 0004h) SPI_CTRLR1

Bits	Access	INI	Symbol	Description
0	R/W	0x0	NDF	<p>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.</p> <p>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.</p>

11.3.3 SPI_SSIENR

- **Name:** SPI SSI Enable Register
- **Size:** 1 bit
- **Initial Value:** 0x0

This register enables and disables the SPI.

REG 11-5 (Offset 0008h) SPI_SSIENR

Bits	Access	INI	Symbol	Description
0	R/W	0x0	SSI_EN	<p>SSI 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.</p>

11.3.4 SPI_MWCR

- **Name:** SPI Microwire Control Register
- **Size:** 3 bits
- **Initial Value:** 0x0

This register controls the direction of the data word for the half-duplex Microwire serial protocol. It is impossible to write to this register when the SPI is enabled.

REG 11-6 (Offset 000Ch) SPI_MWCR

Bits	Access	INI	Symbol	Description
2	R/W	0x0	MHS	<p>Microwire Handshaking. Relevant only when the SPI is configured as a serial-master device. When configured as a serial slave, this bit field has no functionality. Used to enable and disable the “busy/ready” handshaking interface for the Microwire protocol. When enabled, the SPI checks for a ready status from the target slave, after the transfer of the last data/control bit, before clearing the BUSY status in the SR register.</p> <p>0 – handshaking interface is disabled 1 – handshaking interface is enabled</p>
1	R/W	0x0	MDD	<p>Microwire Control. Defines the direction of the data word when the Microwire serial protocol is used. When this bit is set to 0, the data word is received by the SPI from the external serial device. When this bit is set to 1, the data word is transmitted from the SPI to the external serial device.</p>
0	R/W	0x0	MWMOD	<p>Microwire Transfer Mode. Defines whether the Microwire transfer is sequential or non-sequential. When sequential mode is used, only one control word is needed to transmit or receive a block of data words. When non-sequential mode is used, there must be a control word for each data word that is transmitted or received.</p> <p>0 – non-sequential transfer 1 – sequential transfer</p>

11.3.5 SPI_SER

- **Name:** SPI Slave Enable Register
- **Size:** 1 bits
- **Initial Value:** 0x0

This register enables and disables the SPI.

REG 11-7 (Offset 0010h) SPI_SER

Bits	Access	INI	Symbol	Description
0	R/W	0x0	SER	<p>Slave Select Enable Flag. When this bit is set, the slave select line from the master is activated when a serial transfer begins. It should be noted that setting or clearing the bit have no effect on the slave select output until a transfer is started. Before beginning a transfer, you should enable the bit in this register so that SPI master starts to communicate with the slave.</p> <p>1: Selected 0: Not Selected</p>

11.3.6 SPI_BAUDR

- **Name:** SPI Baud Rate Select Register
- **Size:** 16 bits
- **Initial Value:** 0x0

This register defines the ssi_clk divider value. It is valid only when the SPI device is a master and cannot be written when the device is enabled.

REG 11-8 (Offset 0014h) SPI_BAUDR

Bits	Access	INI	Symbol	Description
15:0	R/W	0x0	SCKDV	SSI Clock Divider. If the value is 0, then the sclk_out (SCLK) is disabled. Not recommend to configure this register due to some clock ratio constraints. $f_{ssi_clk} / SCKDV = f_{sclk_out}$

11.3.7 SPI_TXFTLR

- **Name:** SPI Transmit FIFO Threshold Level
- **Size:** 6 bits
- **Initial Value:** 0x0

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 SSINR register.

REG 11-9 (Offset 0018h) SPI_TXFTLR

Bits	Access	INI	Symbol	Description
5:0	R/W	0x0	TFT	Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt. The FIFO depth 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 the table shown below.

Table 11-10 Transmit FIFO Threshold Value Decode Field

TFT Value	Description
00_0000	ssi_txe_intr is asserted when 0 data entries are present in transmit FIFO
00_0001	ssi_txe_intr is asserted when 1 data entries are present in transmit FIFO
00_0010	ssi_txe_intr is asserted when 2 data entries are present in transmit FIFO
:	:
01_1111	ssi_txe_intr is asserted when 63 data entries are present in transmit FIFO

11.3.8 SPI_RXFTLR

- **Name:** SPI Receive FIFO Threshold Level
- **Size:** 6 bits
- **Initial Value:** 0x0

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.

REG 11-11 (Offset 001Ch) SPI_RXFTLR

Bits	Access	INI	Symbol	Description
5:0	R/W	0x0	RFT	<p>Receive FIFO Threshold. Controls the level of entries (or above) at which the receive 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 the depth of the FIFO, this field is not written and retains its current value.</p> <p>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 the table shown below.</p>

Table 11-12 Receive FIFO Threshold Value Decode Field

RFT Value	Description
00_0000	ssi_rxf_intr is asserted when 0 data entries are present in transmit FIFO
00_0001	ssi_rxf_intr is asserted when 1 data entries are present in transmit FIFO
00_0010	ssi_rxf_intr is asserted when 2 data entries are present in transmit FIFO
:	:
01_1111	ssi_rxf_intr is asserted when 63 data entries are present in transmit FIFO

11.3.9 SPI_TXFLR

- **Name:** SPI Transmit FIFO Level Register
- **Size:** 7 bits
- **Initial Value:** 0x0

This register contains the number of valid data entries in the transmit FIFO memory.

REG 11-13 (Offset 0020h) SPI_TXFLR

Bits	Access	INI	Symbol	Description
6:0	R	0x0	TXTFL	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

11.3.10 SPI_RXFLR

- **Name:** SPI Receive FIFO Level Register
- **Size:** 7 bits
- **Initial Value:** 0x0

This register contains the number of valid data entries in the receive FIFO memory. This register can be read at any time.

REG 11-14 (Offset 0024h) SPI_RXFLR

Bits	Access	INI	Symbol	Description
6:0	R	0x0	RXTFL	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO.

11.3.11 SPI_SR

- **Name:** SPI Status Register
- **Size:** 7 bits
- **Initial Value:** 0x06

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 requests an interrupt.

REG 11-15 (Offset 0028h) SPI_SR

Bits	Access	INI	Symbol	Description
6	R	0x0	DCOL	Data Collision Error. Relevant only when the SPI is configured as a master device. This bit is set if the /SS input is asserted by another master, while the SPI master is in the middle of the transfer. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. 0 – No error 1 – Transmission data collision error
5	R	0x0	TXE	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	R	0x0	RFF	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	R	0x0	RFNE	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	R	0x1	TFE	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	R	0x1	TFNF	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	R	0x0	BUSY	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

11.3.12 SPI_IMR

- **Name:** SPI Interrupt Mask Register
- **Size:** 8 bits
- **Initial Value:** 0x3F

This read/write register masks or enables all interrupts generated by the SPI. The MSTIR bit field is not present if the SPI is configured as a serial-slave device. SSRIS and TXUIS are valid when the device is a SPI slave (FRF = 2'b00).

REG 11-16 (Offset 002Ch) SPI_IMR

Bits	Access	INI	Symbol	Description
7	R/W	0x1	SSRIM	SS_N Rising Edge Detect Interrupt Mask. 0 – ssi_ssr_intr interrupt is masked 1 – ssi_ssr_intr interrupt is not masked
6	R/W	0x1	TXUIM	Transmit FIFO Underflow Interrupt Mask. 0 – ssi_txu_intr interrupt is masked 1 – ssi_txu_intr interrupt is not masked
5	R/W	0x1	MSTIM	Multi-Master Contention Interrupt Mask. This bit field is not present if the device is a slave. 0 – ssi_mst_intr interrupt is masked 1 – ssi_mst_intr interrupt is not masked
4	R/W	0x1	RXFIM	Receive FIFO Full Interrupt Mask 0 – ssi_rxf_intr interrupt is masked 1 – ssi_rxf_intr interrupt is not masked
3	R/W	0x1	RXOIM	Receive FIFO Overflow Interrupt Mask 0 – ssi_rxo_intr interrupt is masked 1 – ssi_rxo_intr interrupt is not masked
2	R/W	0x1	RXUIM	Receive FIFO Underflow Interrupt Mask

				0 – ssi_rxu_intr interrupt is masked 1 – ssi_rxu_intr interrupt is not masked
1	R/W	0x1	TXOIM	Transmit FIFO Overflow Interrupt Mask 0 – ssi_txo_intr interrupt is masked 1 – ssi_txo_intr interrupt is not masked
0	R/W	0x1	TXEIM	Transmit FIFO Empty Interrupt Mask 0 – ssi_txe_intr interrupt is masked 1 – ssi_txe_intr interrupt is not masked

11.3.13 SPI_ISR

■ **Name:** SPI Interrupt Status Register

■ **Size:** 8 bits

■ **Initial Value:** 0x0

This register reports the status of the SPI interrupts after they have been masked. The MSTIS bit field is not present if the SPI is configured as a serial-slave device. SSRIS and TXUIS are valid when the device is a SPI slave (FRF = 2'b00).

REG 11-17 (Offset 0030h) SPI_ISR

Bits	Access	INI	Symbol	Description
7	R	0x0	SSRIS	SS_N Rising Edge Detect Interrupt Status. 0 = ssi_ssri_intr interrupt not active after masking 1 = ssi_ssri_intr interrupt is active after masking
6	R	0x0	TXUIS	Transmit FIFO Underflow Interrupt Status. 0 = ssi_txu_intr interrupt not active after masking 1 = ssi_txu_intr interrupt is active after masking
5	R	0x0	MSTIS	Multi-Master Contention Interrupt Status. This bit field is not present if the device is a slave. 0 = ssi_msti_intr interrupt not active after masking 1 = ssi_msti_intr interrupt is active after masking
4	R	0x0	RXFIS	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	R	0x0	RXOIS	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	R	0x0	RXUIS	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	R	0x0	TXOIS	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	R	0x0	TXEIS	Transmit FIFO Empty Interrupt Status 0 = ssi_txe_intr interrupt is not active after masking 1 = ssi_txe_intr interrupt is active after masking

11.3.14 SPI_RISR

- **Name:** SPI Raw Interrupt Status Register
- **Size:** 32 bits
- **Initial Value:** 0x0

This read-only register reports the status of the SPI interrupts prior to masking. The MSTIR bit field is not present if the SPI is configured as a serial-slave device. SSRIS and TXUIS are valid when the device is a SPI slave (FRF = 2'b00).

REG 11-18 (Offset 0034h) SPI_RISR

Bits	Access	INI	Symbol	Description
7	R	0x0	SSRIR	SS_N Rising Edge Detect Raw Interrupt Status. 0 = ssi_ssr_intr interrupt is not active prior to masking 1 = ssi_ssr_intr interrupt is active prior masking
6	R	0x0	TXUIR	Transmit FIFO Underflow Raw Interrupt Status. 0 = ssi_txu_intr interrupt is not active prior to masking 1 = ssi_txu_intr interrupt is active prior masking
5	R	0x0	MSTIR	Multi-Master Contention Raw Interrupt Status. This bit field is not present if the device is a slave. 0 = ssi_mst_intr interrupt is not active prior to masking 1 = ssi_mst_intr interrupt is active prior masking
4	R	0x0	RXFIR	Receive FIFO Full Raw Interrupt Status 0 = ssi_rxf_intr interrupt is not active prior to masking 1 = ssi_rxf_intr interrupt is full prior to masking
3	R	0x0	RXOIR	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	R	0x0	RXUIR	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	R	0x0	TXOIR	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	R	0x0	TXEIR	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

11.3.15 SPI_TXOICR

- **Name:** SPI Transmit FIFO Overflow Interrupt Clear Register
- **Size:** 1 bit
- **Initial Value:** 0x0

REG 11-19 (Offset 0038h) SPI_TXOICR

Bits	Access	INI	Symbol	Description
0	R	0x0	TXOICR	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.

11.3.16 SPI_RXOICR

- **Name:** SPI Receive FIFO Overflow Interrupt Clear Register
- **Size:** 1 bit
- **Initial Value:** 0x0

REG 11-20 (Offset 003Ch) SPI_RXOICR

Bits	Access	INI	Symbol	Description
0	R	0x0	RXOICR	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.

11.3.17 SPI_RXUICR

- **Name:** SPI Receive FIFO Underflow Interrupt Clear Register
- **Size:** 1 bit
- **Initial Value:** 0x0

REG 11-21 (Offset 0040h) SPI_RXUICR

Bits	Access	INI	Symbol	Description
0	R	0x0	RXUICR	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.

11.3.18 SPI_MSTICR

- **Name:** SPI Multi-Master Interrupt Clear Register
- **Size:** 1 bit
- **Initial Value:** 0x0

REG 11-22 (Offset 0044h) SPI_MSTICR

Bits	Access	INI	Symbol	Description
0	R	0x0	MSTICR	Clear Multi-Master Contention Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_mst_intr interrupt; writing has no effect. This register is available when SPI is master.

11.3.19 SPI_ICR

- **Name:** SPI Interrupt Clear Register
- **Size:** 1 bit
- **Initial Value:** 0x0

REG 11-23 (Offset 0048h) SPI_ICR

Bits	Access	INI	Symbol	Description
0	R	0x0	ICR	Clear Interrupts. This register is set if any of the interrupts below are active. A read clears the ssi_txu_intr, ssi_txo_intr, ssi_rxu_intr, ssi_rxo_intr, ssi_ssri_intr, ssi_txu_intr and the ssi_mst_intr interrupts. Writing to this register has no effect.

11.3.20 SPI_DMCR

- **Name:** SPI DMA Control Register
- **Size:** 2 bits
- **Initial Value:** 0x0

Enable SPI handshake interface with DMA.

REG 11-24 (Offset 004Ch) SPI_DMCR

Bits	Access	INI	Symbol	Description
1	R/W	0x0	TDMAE	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. 0 = Transmit DMA disabled 1 = Transmit DMA enabled
0	R/W	0x0	RDMAE	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel 0 = Receive DMA disabled 1 = Receive DMA enabled

11.3.21 SPI_DMATDLR

- **Name:** SPI DMA Transmit Data Level
- **Size:** 6 bits
- **Initial Value:** 0x0

REG 11-25 (Offset 0050h) SPI_DMATDLR

Bits	Access	INI	Symbol	Description
5:0	R/W	0x0	DMATDL	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. The SPI generates a request signal to the DMA engine when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1. This value has been optimized to the watermark level, which is associated with FIFO depth and burst transaction size. User should not modify this value to prevent unexpected errors.

11.3.22 SPI_DMARDLR

- **Name:** SPI DMA Receive Data Level
- **Size:** 6 bits
- **Initial Value:** 0x0

REG 11-26 (Offset 0054h) SPI_DMARDLR

Bits	Access	INI	Symbol	Description
5:0	R/W	0x0	DMARDL	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The SPI generate a DMA request signal when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE=1. This value associated with FIFO depth and burst transaction size has been optimized. User should not modify this value to prevent unexpected errors.

11.3.23 SPI_TXUICR

- **Name:** SPI Transmit FIFO Underflow Interrupt Clear Register
- **Size:** 1 bit
- **Initial Value:** 0x0

REG 11-27 (Offset 0058h) SPI_TXUICR

Bits	Access	INI	Symbol	Description
0	R	0x0	TXUICR	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.

11.3.24 SPI_SSICR

- **Name:** SPI SS_N Rising Edge Detect Interrupt Clear Register
- **Size:** 1 bit
- **Initial Value:** 0x0

REG 11-28 (Offset 005Ch) SPI_SSICR

Bits	Access	INI	Symbol	Description
0	R	0x0	SSICR	Clear SS_N Rising Edge Detect Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_ss_intr interrupt; writing has no effect.

11.3.25 SPI_DR

- **Name:** SPI Data Register
- **Size:** 16 bits
- **Initial Value:** 0x0

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 and cleared when SSI_EN = 0. The depth of FIFO is 64.

REG 11-29 (Offset 0060h) SPI_DR

Bits	Access	INI	Symbol	Description
15:0	R/W	0x0	DR	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

12 Inter-Integrated Circuit Interface (I2C)

12.1 Overview

12.1.1 Application Scenario

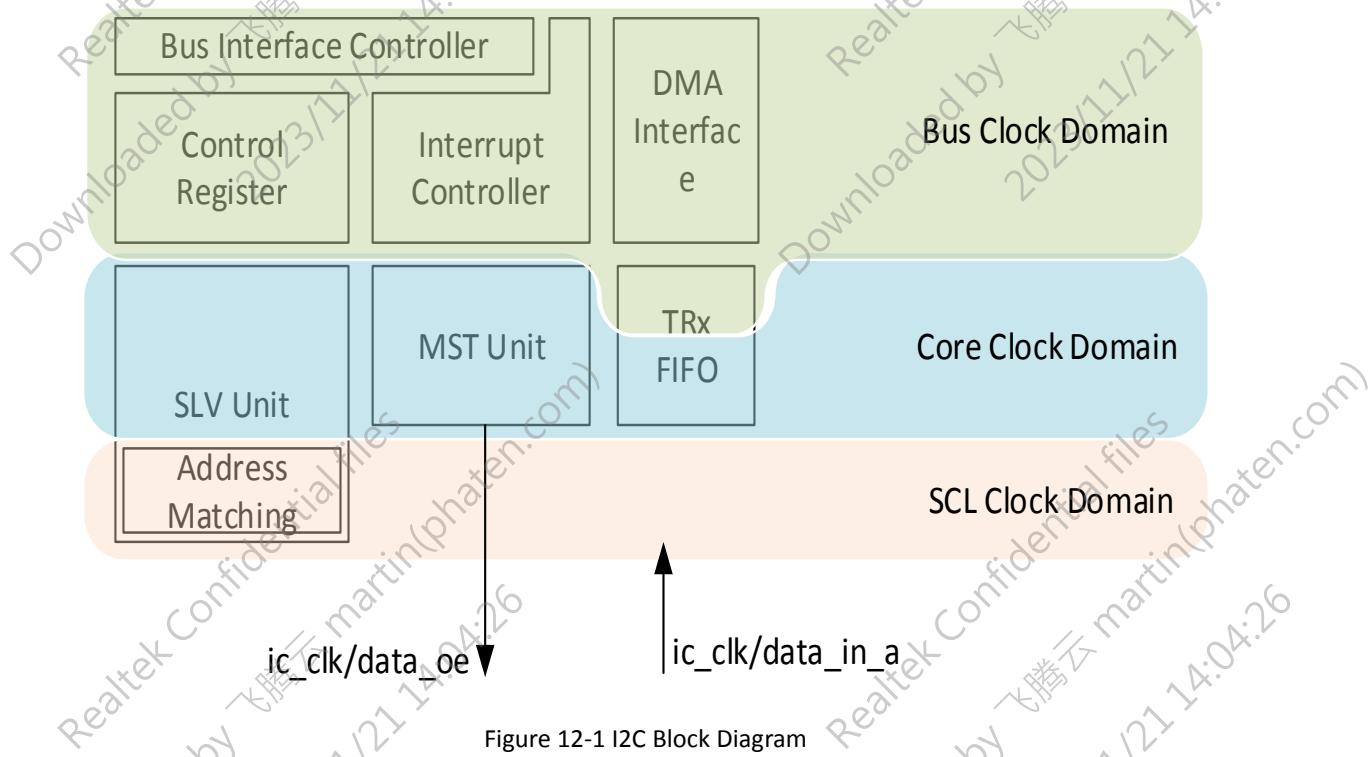
The design of I2C aims on sensor hub application in low-power or battery-powered productions. Essential features of I2C bus protocol should be provided for acquiring or controlling external sensor data. To reduce system power consumption, advanced application scheme in Ameba-Z II I2C are also available for further low power system state. For I2C master mode, a DMA-based auto read/write would drive bus command according to the DMA source register configuration. For I2C slave mode, I2C could set an interrupt signal to wake up CPU only when I2C address matches its own slave address.

12.1.2 Features

- Communication speed
 - Standard speed (up to 100 kHz)
 - Fast speed (up to 400 kHz)
- Address mode
 - 7-bit and 10-bit
- Master mode
 - Multi-master ability including bus arbitration scheme
 - Manual STOP/RESTART bit control
 - Auto read/write DMA mode
 - Connection Status command (Only slave address been sent)
- Slave mode
 - Dual own address. The 2nd slave address only can be 7-bit.
 - Address matching wake up
- Clock stretch in master/slave mode
- More flexible DMA and FIFO architecture
 - 8-bit and 16-bit (DWC legacy) width FIFO
 - Byte DMA support
 - TX/RX FIFO with 32 entries

12.1.3 Architecture

The Ameba-Z II I2C is made up of an AMBA APB slave interface, an I2C interface, and FIFO logic to maintain coherency between the two interfaces. A simplified block diagram of the component is illustrated in Figure 12-1.



12.1.3.1 Functional Block Diagram with Clock Domain

- Bus interface controller: Takes the bus interface signals and translates them into a common generic interface that allows the register file to be bus protocol-agnostic.
- Control register: Contains configuration registers and is the interface with software.
- SLV Unit: Follows the protocol for a slave and monitors bus for address match.
- MST Unit: Generates the I2C protocol for the master transfers.
- Interrupt controller: Generates the raw interrupt and interrupt flags, allowing them to be set and cleared.
- DMA interface: Generates the handshaking signals to the central DMA controller in order to automate the data transfer without CPU intervention.
- TRx FIFO: Holds the RX FIFO and TX FIFO register banks and controllers, along with their status levels.
- Address Matching: Slave address match process

12.2 Functional Description

Ameba-Z II I2C basically adopts general I2C design architecture of DWC I2C and provides extended features for reducing CPU computing and power consumption. Several different operation types could be applied to various system architectures. Direct IO, interrupt, 16-bit FIFO (DWC legacy), 8-bit FIFO with transfer descriptor parse and 8-bit FIFO with transfer control register would be sufficient for different application requirements.

12.2.1 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.

12.2.2 I2C Terminology

The following terms are used throughout this manual and are defined as follows:

12.2.2.1 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 below:

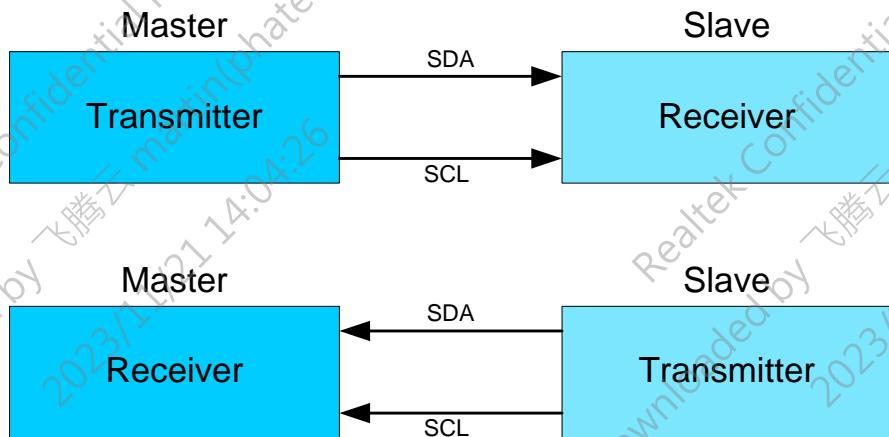


Figure 12-2 I2C Master/Slave and Transmitter/Receiver Relationship

- **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)

12.2.2.2 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.
- **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.

12.2.3 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 or 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 below.

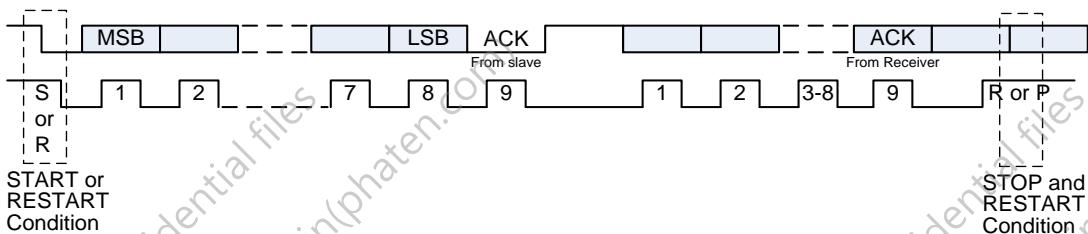


Figure 12-3 Data Transfer on The I2C Bus

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 400 pF. Data is transmitted in byte packages.

12.2.3.1 START and STOP Generation

When operating as an I2C master, putting data into the transmit FIFO causes the I2C to generate a START condition on the I2C bus. Writing a 1 to REG_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 REG_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-Z II I2C does not generate START and STOP conditions, as per the protocol. However, if a read request is made to the Ameba-Z II 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-Z II I2C, or the Ameba-Z II I2C slave is disabled by writing a 0 to bit0 of the REG_IC_ENABLE.

12.2.3.2 Combined Formats

The Ameba-Z II 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, REG_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:

- REG_IC_DATA_CMD[9] is checked and:
 - If set to 1, a STOP bit is issued.
 - If set to 0, the SCL is held low until the next command is written to the transmit FIFO.

12.2.4 I2C Protocols

The I2C has the protocols discussed in this section.

12.2.4.1 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. Figure 12-4 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.

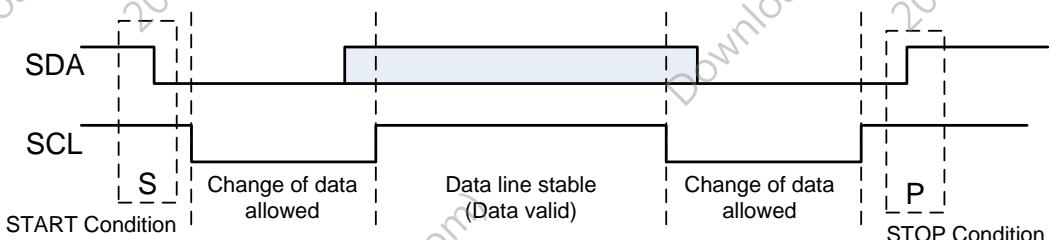


Figure 12-4 START and STOP Conditions

12.2.4.2 Addressing Slave Protocol

There are two address formats: the 7-bit address format and the 10-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 Figure 12-5. 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.

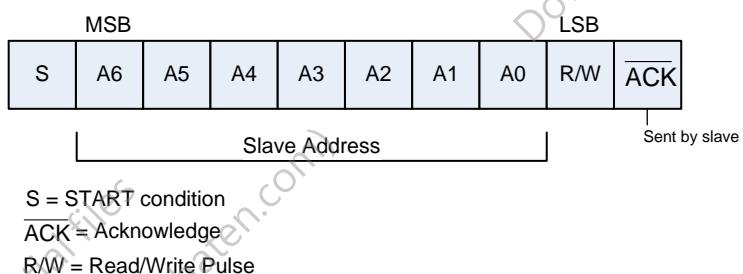
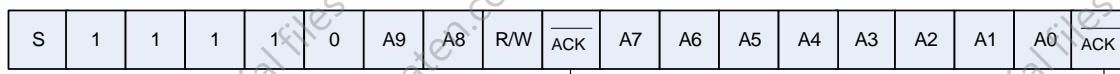


Figure 12-5 7-Bit Address Format

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. Figure 12-6 shows the 10-bit address format.



S = START condition
ACK = Acknowledge
R/W = Read/Write Pulse

Figure 12-6 10-Bit Address Format

12.2.4.3 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.

12.2.4.3.1 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 Figure 12-7, then the slave-receiver responds to the master-transmitter with an acknowledge pulse after every byte of data is received.

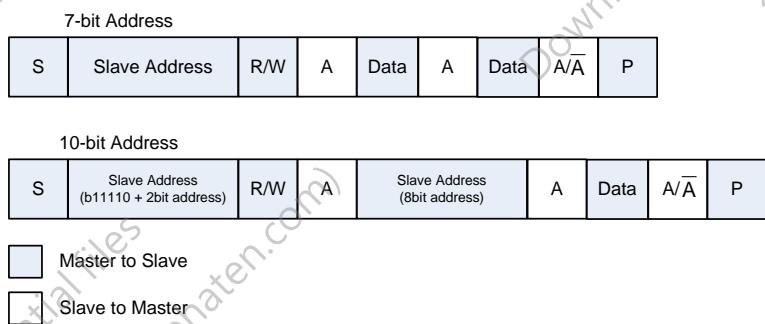


Figure 12-7 Master-Transmitter Protocol

12.2.4.3.2 Master-Receiver and Slave-Transmitter

If the master is receiving data as shown in Figure 12-8, 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 Ameba-Z II I2C can then communicate with the same slave using a transfer of a different direction.

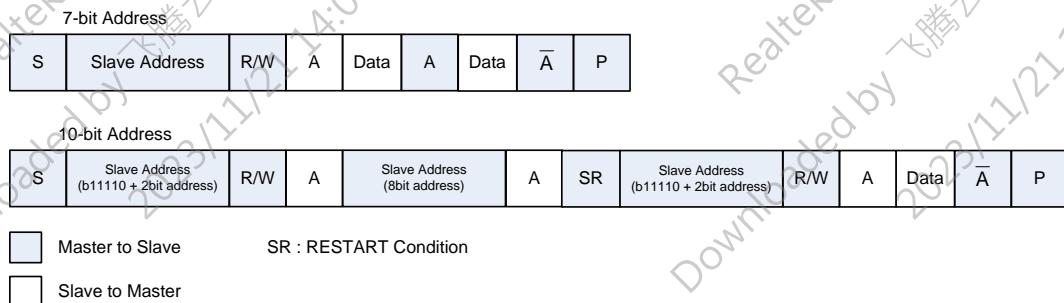


Figure 12-8 Master-Receiver (Slave-Transmitter) Protocol

12.2.5 Tx FIFO Operation and Control

Ameba-Z II's 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. Figure 12-9 shows the fields in REG_IC_DATA_CMD. Please refer to the register description for more detail information.

REG_IC_DATA_CMD	Reserved	NULL_DAT	RESTART	STOP	CMD	DATA	0
15	12	11	10	9	8	7	0

Figure 12-9 IC_DATA_CMD Register Content

Figure 12-10 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.

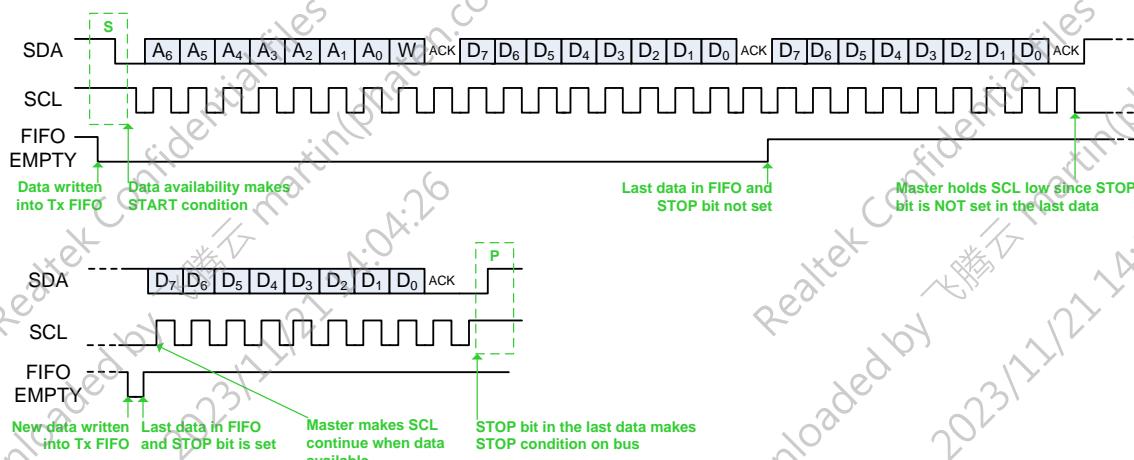


Figure 12-10 Master Transmitter – TX FIFO Empty and STOP Generation

Figure 12-11 illustrates the behavior of the I2C when the TX FIFO becomes empty while operating as a master receiver.

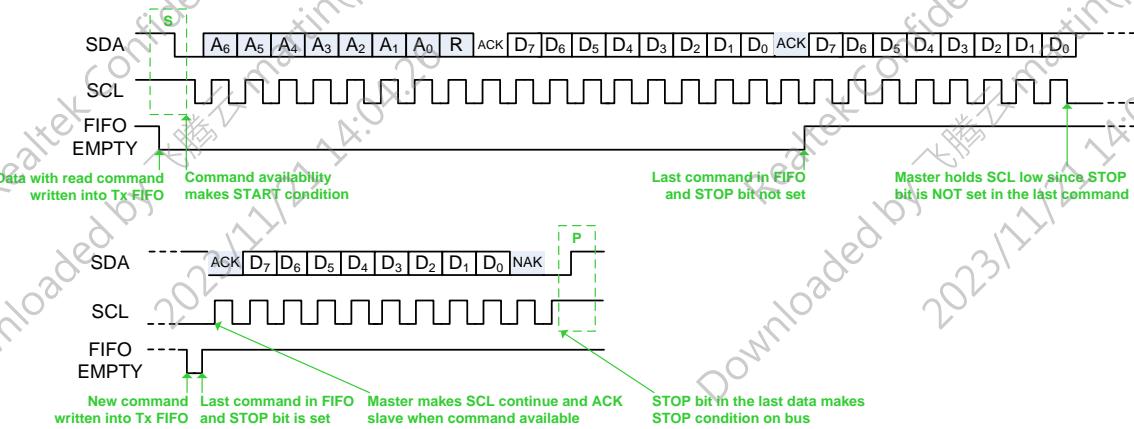


Figure 12-11 Master Receiver – Tx FIFO Empty and STOP Generation

Figure 12-12 and Figure 12-13 illustrate configurations where the user can control the generation of RESTART conditions on the I2C bus. If bit 10 (RESTART) of the REG_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. Figure 12-12 illustrates this situation during operation as a master transmitter.

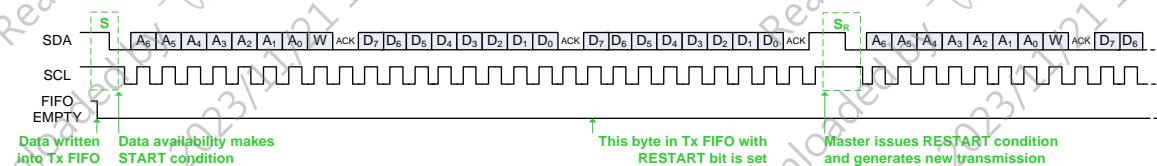


Figure 12-12 Master Transmitter – RESTART Generation

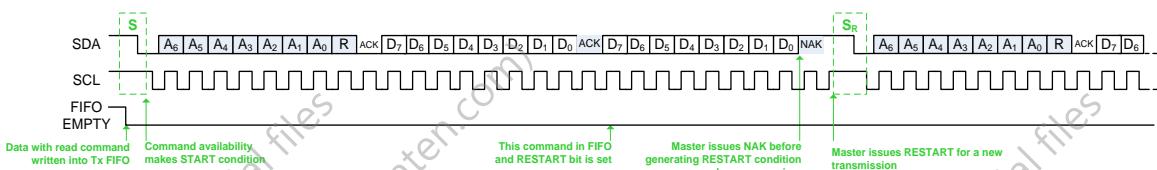


Figure 12-13 Master Receiver – RESTART Generation

Figure 12-14 illustrates operation as a master transmitter where the Stop bit of the REG_IC_DATA_CMD register is set and the Tx FIFO is not empty.

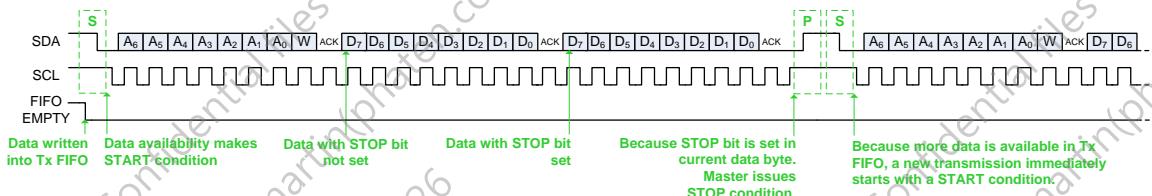
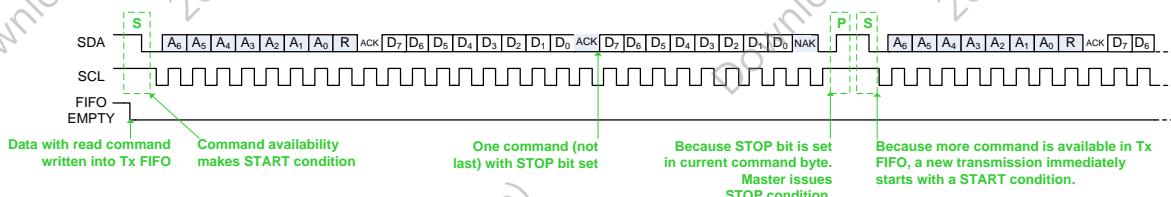


Figure 12-15 illustrates operation as a master receiver where the STOP bit of the REG_IC_DATA_CMD register is set and the Tx FIFO is not empty.



12.2.5.1 DMA Operation

12.2.5.1.1 DWC Legacy Control Mode

Refer to the DWC databook. Direct IO, interrupt and 16-bit FIFO mode would be introduced in the databook. This mode is NOT recommended to use in application.

12.2.5.1.2 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 DMAC 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.

In 8-bit FIFO with DMA transfer control register mode, transfer process is controlled by two additional register: DMA transfer control and DMA transfer data length 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.

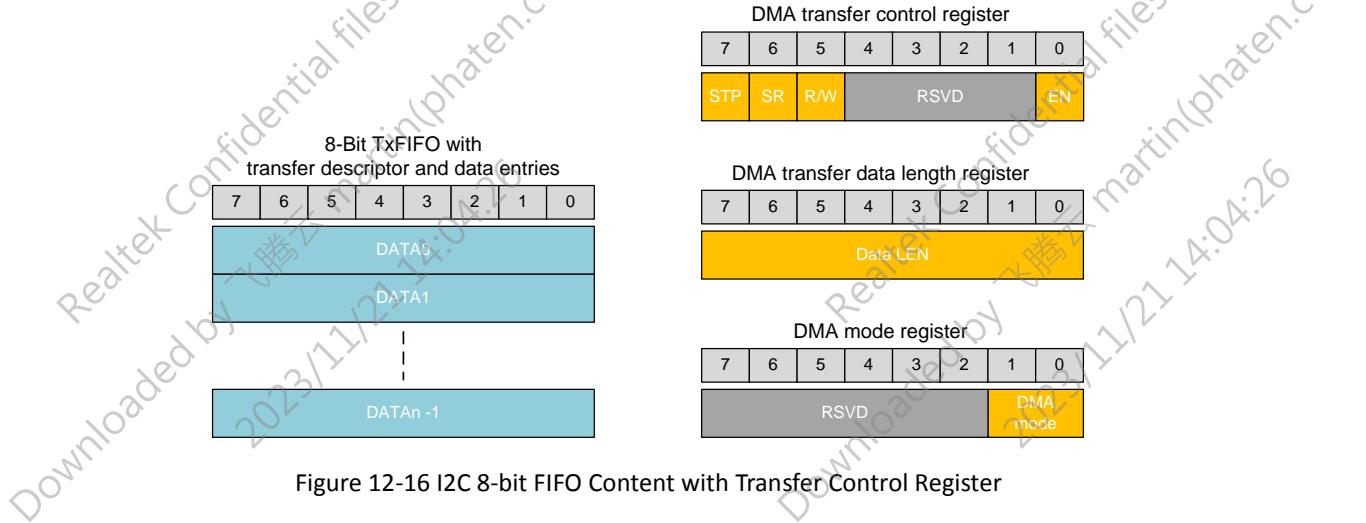


Figure 12-16 I2C 8-bit FIFO Content with Transfer Control Register

12.3 Control Registers

12.3.1 IC_CON

- Name: I2C Control Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-1 (Offset 0000h) IC_CON

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7	R/W	0	IC_CON_IC_SLAVE_DISABLE_1	This bit controls whether I2C has its slave1 disabled, which means once the presetn signal is applied, then this bit takes on the value of the configuration parameter IC_SLAVE_DISABLE_1. If this bit is set (slave is disabled), I2C Module functions only as a master and does not perform any action that requires a slave.
6	R/W	0	IC_CON_IC_SLAVE_DISABLE	This bit controls whether I2C has its slave disabled, which means once the presetn signal is applied, then this bit takes on the value of the configuration parameter IC_SLAVE_DISABLE. If this bit is set (slave is disabled), I2C Module functions only as a master and does not perform any action that requires a slave.
5	R/W	0	IC_CON_IC_RESTART_EN	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 Module operations.
4	R/W	0	IC_CON_IC_10BITADDR_MASTER	Controls whether the I2C Module starts its transfers in 7- or 10-bit addressing mode when acting as a master. The function of this bit is replaced by IC_TAR_IC_10BIT_ADDR (bit12 of REG_IC_TAR.)
3	R/W	0	IC_CON_IC_10BITADDR_SLAVE	When acting as a slave, this bit controls whether the I2C Module responds to 7- or 10-bit addresses.

2:1	R/W	0	IC_CON_SPEED	These bits control at which speed the I2C Module operates; its setting is relevant only if one is operating the I2C Module in master mode.
0	R/W	0	IC_CON_MASTER_MODE	This bit controls whether the I2C Module master is enabled.

12.3.2 IC_TAR

- Name: I2C Target Address Register
- Width: 32bit
- Initial Value: 0x0010

REG 12-2 (Offset 0004h) IC_TAR

Bit	Access	INI	Symbol	Description
31:13	R	0	RSVD	
12	R/W	0	IC_TAR_IC_10BIT_ADDR	This bit controls whether the I2C Module starts its transfers in 7-or 10-bit addressing mode when acting as a master.
11	R/W	0	IC_TAR_SPECIAL	This bit indicates whether software performs a General Call or START BYTE command.
10	R/W	0	IC_TAR_GC_OR_START	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 I2C Module.
9:0	R/W	10	IC_TAR	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.

12.3.3 IC_SAR

- Name: I2C Slave Address Register
- Width: 32bit
- Initial Value: 0x0011

REG 12-3 (Offset 0008h) IC_SAR

Bit	Access	INI	Symbol	Description
31:10	R	0	RSVD	
9:0	R/W	11	IC_SAR	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. 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.

12.3.4 IC_HS_MADDR

- Name: I2C HS Master Mode Code Address Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-4 (Offset 000Ch) IC_HS_MADDR

Bit	Access	INI	Symbol	Description
31:3	R	0	RSVD	
2:0	R/W	0	IC_HS_MADDR	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 highspeed mode masters can be present on the same I2C bus system. Valid values are from 0 to 7.

12.3.5 IC_DATA_CMD

- Name: I2C Slave 0 and Master Rx/Tx Data Buffer Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-5 (Offset 0010h) IC_DATA_CMD

Bit	Access	INI	Symbol	Description
31:12	R	0	RSVD	
11	W	0	IC_DATA_CMD_NULL_DATA	([11:8] exist if DMA_MODE= 0) This bit controls whether to transfer slave address only. When this bit 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. 1: Send TXFIFO data as slave address. 0: Normal operation. This bit will NOT influence any transfer sequence.
10	W	0	IC_DATA_CMD_RESTART	This bit controls whether a RESTART is issued before the next byte is sent or received.
9	W	0	IC_DATA_CMD_STOP	This bit controls whether a STOP is issued after the byte is sent or received.
8	W	0	IC_DATA_CMD_CMD	This bit controls whether a read or a write is performed. This bit does not control the direction when the I2C Module acts as a slave. It controls only the direction when it acts as a master.
7:0	R/W	0	IC_DATA_CMD_DAT	This register contains the data to be transmitted or received on the I2C bus. If you are writing to this register and want to perform a read, bits 7:0 (DAT) are ignored by the I2C Module. However, when you read this register, these bits return the value of data received on the I2C Module interface.

12.3.6 IC_SS_SCL_HCNT

- Name: Standard Speed I2C Clock SCL High Count Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-6 (Offset 0014h) IC_SS_SCL_HCNT

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R/W	190	IC_SS_SCL_HCNT	This register sets the SCL clock high-period count for standard speed.

12.3.7 IC_SS_SCL_LCNT

- Name: Standard Speed I2C Clock SCL Low Count Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-7 (Offset 0018h) IC_SS_SCL_LCNT

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R/W	1D6	IC_SS_SCL_LCNT	This register sets the SCL clock low period count for standard speed.

12.3.8 IC_FS_SCL_HCNT

- Name: Fast Mode or Fast Mode Plus I2C Clock SCL High Count Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-8 (Offset 001Ch) IC_FS_SCL_HCNT

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R/W	3C	IC_FS_SCL_HCNT	This register sets the SCL clock high-period count for fast speed.

12.3.9 IC_FS_SCL_LCNT

- Name: Fast Mode or Fast Mode Plus I2C Clock SCL Low Count Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-9 (Offset 0020h) IC_FS_SCL_HCNT

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R/W	82	IC_FS_SCL_LCNT	This register sets the SCL clock low period count for fast speed.

12.3.10 IC_HS_SCL_HCNT

- Name: High Speed I2C Clock SCL High Count Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-10 (Offset 0024h) IC_HS_SCL_HCNT

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R/W	6	IC_HS_SCL_HCNT	This register sets the SCL clock high period count for high speed.

12.3.11 IC_HS_SCL_LCNT

- Name: High Speed I2C Clock SCL Low Count Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-11 (Offset 0028h) IC_HS_SCL_LCNT

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R/W	6	IC_HS_SCL_HCNT	This register sets the SCL clock low period count for high speed.

12.3.12 IC_INTR_STAT

- Name: I2C Interrupt Status Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-12 (Offset 002Ch) IC_INTR_STAT

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R	0	IC_INTR_DMA_I2C_DONE	(Active if IC_DMA_MODE = 1)
14	R	0	IC_INTR_MS_CODE_DET	(Active in HS mode)
13	R	0	IC_INTR_ADDR_2_MATCH	(Active w/wo clock src)
12	R	0	IC_INTR_ADDR_1_MATCH	(Active w/wo clock src)
11	R	0	IC_INTR_STAT_R_GEN_CALL	
10	R	0	IC_INTR_STAT_R_START_DET	
9	R	0	IC_INTR_STAT_R_STOP_DET	
8	R	0	IC_INTR_STAT_R_ACTIVITY	
7	R	0	IC_INTR_STAT_R_RX_DONE	
6	R	0	IC_INTR_STAT_R_TX_ABRT	
5	R	0	IC_INTR_STAT_R_RD_REQ	
4	R	0	IC_INTR_STAT_R_TX_EMPTY	
3	R	0	IC_INTR_STAT_R_TX_OVER	
2	R	0	IC_INTR_STAT_R_RX_FULL	
1	R	0	IC_INTR_STAT_R_RX_OVER	
0	R	0	IC_INTR_STAT_R_RX_UNDER	

12.3.13 IC_INTR_MASK

- Name: I2C Interrupt Mask Register
- Width: 32bit
- Initial Value: 0xFFFF

REG 12-13 (Offset 0030h) IC_INTR_MASK

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	IC_INTR_MASK_M_DMA_I2C_DONE	
14	R/W	0	IC_INTR_MASK_M_MS_CODE_DET	
13	R/W	0	IC_INTR_MASK_M_ADDR_2_MATCH	(active only if IC_SLAVE_DISABLE_1 is 0)
12	R/W	0	IC_INTR_MASK_M_ADDR_1_MATCH	
11	R/W	0	IC_INTR_MASK_M_GEN_CALL	
10	R/W	0	IC_INTR_MASK_M_START_DET	

9	R/W	0	IC_INTR_MASK_M_STOP_DET
8	R/W	0	IC_INTR_MASK_M_ACTIVITY
7	R/W	0	IC_INTR_MASK_M_RX_DONE
6	R/W	0	IC_INTR_MASK_M_TX_ABRT
5	R/W	0	IC_INTR_MASK_M_RD_REQ
4	R/W	0	IC_INTR_MASK_M_TX_EMPTY
3	R/W	0	IC_INTR_MASK_M_TX_OVER
2	R/W	0	IC_INTR_MASK_M_RX_FULL
1	R/W	0	IC_INTR_MASK_M_RX_OVER
0	R/W	0	IC_INTR_MASK_M_RX_UNDER

12.3.14 IC_RAW_INTR_STAT

- Name: I2C Raw Interrupt Status Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-14 (Offset 0034h) IC_RAW_INTR_STAT

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R	0	IC_RAWINTR_DMA_I2C_DONE	Indicates whether I2C DMA operation is done.
14	R	0	IC_RAW_INTR_MS_CODE_DET	Indicates whether master code is detected.
13	R	0	RAW_IC_INTR_ADDR_2_MATC_H	Indicates whether an address matches with slave address 1 when acting as a slave.
12	R	0	IC_RAW_INTR_ADDR_1_MATC_H	Indicates whether an address matches with slave address 0 when acting as a slave.
11	R	0	IC_RAW_INTR_STAT_GEN_CALL	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling I2C Module or when the CPU reads bit 0 of the REG_DW_I2C_IC_CLR_GEN_CALL register. The received data is stored in the Rx buffer.
10	R	0	IC_RAW_INTR_STAT_START_DET	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether I2C Module is operating in slave or master mode.
9	R	0	IC_RAW_INTR_STAT_STOP_DET	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether I2C Module is operating in slave or master mode.
8	R	0	IC_RAW_INTR_STAT_ACTIVITY	This bit captures I2C Module activity and stays set until it is cleared. There are four ways to clear it:
7	R	0	IC_RAW_INTR_STAT_RX_DONE	When the I2C Module 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	R	0	IC_RAW_INTR_STAT_TX_ABRT	This bit indicates if I2C Module, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a "transmit abort". When this bit is set to 1, the REG_DW_I2C_IC_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes places

5	R	0	IC_RAW_INTR_STAT_RD_REQ	This bit is set to 1 when I2C Module is acting as a slave and another I2C master is attempting to read data from I2C Module. The I2C Module holds the I2C 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 REG_DW_I2C_IC_DATA_CMD register. This bit is set to 0 just after the processor reads the REG_DW_I2C_IC_CLR_RD_REQ register.
4	R	0	IC_RAW_INTR_STAT_TX_EMPTY	This bit is set to 1 when the transmit buffer is at or below the threshold value set in the REG_DW_I2C_IC_TX_TL register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the REG_DW_I2C_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	R	0	IC_RAW_INTR_STAT_TX_OVER	Set during transmit if the transmit buffer is filled to IC_RX_BUFFER_DEPTH and
2	R	0	IC_RAW_INTR_STAT_RX_FULL	Set when the receive buffer reaches or goes above the RX_TL threshold in the REG_DW_I2C_IC_RX_TL register. It is automatically cleared by hardware when buffer level goes
1	R	0	IC_RAW_INTR_STAT_RX_OVER	Set if the receive buffer is completely filled to IC_RX_BUFFER_DEPTH and an
0	R	0	IC_RAW_INTR_STAT_RX_UNDER	Set if the processor attempts to read the receive buffer when it is empty by reading from the REG_DW_I2C_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.

12.3.15 IC_RX_TL

- Name: I2C Receive FIFO Threshold Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-15 (Offset 0038h) IC_RX_TL

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7:0	R/W	B	IC_RX_TL	Receive FIFO Threshold Level

12.3.16 IC_TX_TL

- Name: I2C Transmit FIFO Threshold Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-16 (Offset 003Ch) IC_TX_TL

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	
7:0	R/W	B	IC_TX_TL	Transmit FIFO Threshold Level

12.3.17 IC_CLR_INTR

- Name: I2C Clear Combined and Individual Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-17 (Offset 0040h) IC_CLR_INTR

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	IC_CLR_INTR	Read this register to clear the combined interrupt, all individual interrupts, and the REG_DW_I2C_IC_TX_ABRT_SOURCE register. This bit does not clear hardware clearable interrupts but software clearable interrupts. Refer to Bit 9 of the REG_DW_I2C_IC_TX_ABRT_SOURCE register for an exception to clearing REG_DW_I2C_IC_TX_ABRT_SOURCE.

12.3.18 IC_CLR_RX_UNDER

- Name: I2C Clear RX_UNDER Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-18 (Offset 0044h) IC_CLR_RX_UNDER

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	IC_CLR_RX_UNDER	Read this register to clear the RX_UNDER interrupt (bit 0) of the REG_IC_RAW_INTR_STAT

12.3.19 IC_CLR_RX_OVER

- Name: I2C Clear RX_OVER Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-19 (Offset 0048h) IC_CLR_RX_OVER

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	IC_CLR_RX_OVER	Read this register to clear the RX_OVER interrupt (bit 1) of the REG_IC_RAW_INTR_STAT

12.3.20 IC_CLR_TX_OVER

- Name: I2C Clear TX_OVER Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-20 (Offset 004Ch) IC_CLR_TX_OVER

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	IC_CLR_TX_OVER	Read this register to clear the TX_OVER interrupt (bit 3) of the REG_IC_RAW_INTR_STAT

12.3.21 IC_CLR_RD_REQ

- Name: I2C Clear RD_REQ Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-21 (Offset 0050h) IC_CLR_RD_REQ

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	IC_CLR_RD_REQ	Read this register to clear the RD_REQ interrupt (bit 5) of the REG_IC_RAW_INTR_STAT

12.3.22 IC_CLR_TX_ABRT

- Name: I2C Clear TX_ABRT Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-22 (Offset 0054h) IC_CLR_TX_ABRT

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	CLR_TX_ABRT	Read this register to clear the TX_ABRT interrupt (bit 6) of the REG_DW_I2C_IC_RAW_INTR_STAT register, and the REG_DW_I2C_IC_TX_ABRT_SOURCE register. This also releases the TX FIFO from the flushed/reset state, allowing more writes to the TX FIFO. Refer to Bit 9 of the REG_DW_I2C_IC_TX_ABRT_SOURCE register for an exception to clearing REG_IC_TX_ABRT_SOURCE

12.3.23 IC_CLR_RX_DONE

- Name: I2C Clear RX_DONE Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-23 (Offset 0058h) IC_CLR_RX_DONE

Bit	Access	INI	Symbol	Description
31:1	R/	0	RSVD	
0	R	0	IC_CLR_RX_DONE	Read this register to clear the RX_DONE interrupt (bit 7) of the REG_IC_RAW_INTR_STAT

12.3.24 IC_CLR_ACTIVITY

- Name: I2C Clear ACTIVITY Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-24 (Offset 005Ch) IC_CLR_ACTIVITY

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	IC_CLR_ACTIVITY	Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY 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 ACTIVITY interrupt (bit 8) of the REG_DW_I2C_IC_RAW_INTR_STAT register.

12.3.25 IC_CLR_STOP_DET

- Name: I2C Clear STOP_DET Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-25 (Offset 0060h) IC_CLR_STOP_DET

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	IC_CLR_STOP_DET	Read this register to clear the STOP_DET interrupt (bit 9) of the REG_IC_RAW_INTR_STAT

12.3.26 IC_CLR_START_DET

- Name: I2C Clear START_DET Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-26 (Offset 0064h) IC_CLR_START_DET

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	IC_CLR_START_DET	Read this register to clear the START_DET interrupt (bit 10) of the REG_IC_RAW_INTR_STAT

12.3.27 IC_CLR_GEN_CALL

- Name: I2C Clear GEN_CALL Interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-27 (Offset 0068h) IC_CLR_GEN_CALL

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	
0	R	0	IC_CLR_GEN_CALL	Read this register to clear the GEN_CALL interrupt (bit 11) of the REG_IC_RAW_INTR_STAT

12.3.28 IC_ENABLE

- Name: I2C Enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-28 (Offset 006Ch) IC_ENABLE

Bit	Access	INI	Symbol	Description
31:2	R	0	RSVD	
1	R/W	0	IC_ABORT	Abort I2C current transfer is done w/o flush Tx/Rx FIFO
0	R/W	0	IC_ENABLE	Controls whether the I2C Module is enabled.

12.3.29 IC_STATUS

- Name: I2C Status Register
- Width: 32bit
- Initial Value: 0x0006

REG 12-29 (Offset 0070h) IC_STATUS

Bit	Access	INI	Symbol	Description
31:13	R	0	RSVD	
12:11	R	0	IC_BUS_STATUS	Bus status. 00 indicates I2C bus is in idle 01: I2C bus is in address phase 10: I2C bus is in data phase 11: I2C bus is in clock stretch phase
10	R	0	IC_STATUS_SLV_HOLD_RX_FIFO_FULL	1: I2C hardware is holding I2C bus low (clock stretch) because of RX FIFO full in slave mode.
9	R	0	IC_STATUS_SLV_HOLD_TX_FIFO_EMPTY	1: I2C hardware is holding I2C bus low (clock stretch) because of TX FIFO empty in slave mode.
8	R	0	IC_STATUS_MST_HOLD_RX_FIFO_O_FULL	1: I2C hardware is holding I2C bus low (clock stretch) because of RX FIFO full in master mode.
7	R	0	IC_STATUS_MST_HOLD_TX_FIFO_O_EMPTY	1: I2C hardware is holding I2C bus low (clock stretch) because of TX FIFO empty in master mode.
6	R	0	IC_STATUS_SLV_ACTIVITY	Slave FSM Activity Status. When the Slave Finite State Machine (FSM) is not in the IDLE state, this bit is set.
5	R	0	IC_STATUS_MST_ACTIVITY	Master FSM Activity Status. When the Master Finite State Machine (FSM) is not in the IDLE state, this bit is set.
4	R	0	IC_STATUS_RFF	Receive FIFO CompletelyFull. When the receive FIFO iscompletely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared.
3	R	0	IC_STATUS_RFNE	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.
2	R	1	IC_STATUS_TFE	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.
1	R	1	IC_STATUS_TFNF	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full.
0	R	0	IC_STATUS_ACTIVITY	I2C Activity Status.

12.3.30 IC_TXFLR

- Name: I2C Transmit FIFO Level Register
- Width: 32bit, Available Size: TX_ABW + 1, TX_ABW is 5
- Initial Value: 0x0000

REG 12-30 (Offset 0074h) IC_TXFLR

Bit	Access	INI	Symbol	Description
31:6	R	0	RSVD	RSVD.
5:0	R	0	TXFLR	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

12.3.31 IC_RXFLR

- Name: I2C Receive FIFO Level Register
- Width: 32bit, Real Size: RX_ABW + 1, RX_ABW is 4
- Initial Value: 0x0000

REG 12-31 (Offset 0078h) IC_RXFLR

Bit	Access	INI	Symbol	Description
31:5	R	0	RSVD	RSVD.
4:0	R	0	RXFLR	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO.

12.3.32 IC_SDA_HOLD

- Name: I2C SDA Hold Time Length Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-32 (Offset 007Ch) IC_SDA_HOLD

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	RSVD.
15:0	R/W	16'h0001	IC_SDA_TX_HOLD	Sets the required SDA hold time in units of ic_clk period, when DW_apb_I2C acts as a transmitter.

12.3.33 IC_TX_ABRT_SOURCE

- Name: I2C Transmit Abort Source Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-33 (Offset 0080h) IC_TX_ABRT_SOURCE

Bit	Access	INI	Symbol	Description
31:23	R	0	TX_FLUSH_CNT	This field indicates the number of Tx FIFO data commands that are flushed due to TX_ABRT interrupt. It is cleared whenever I2C is disabled. Role of Realtek I2C: Master-Transmitter or Slave-Transmitter
22:17	R	0	RSVD	RSVD
16	R	0	ABRT_USER_ABRT	This is a master-mode-only bit. Master has detected the transfer abort (IC_ENABLE[1]). Role of Realtek I2C: Master-Transmitter
15	R	0	ABRT_SLVRD_INTX	1: When the processor side responds to a slave mode request for data to be transmitted to a remote master and user writes a 1 in CMD(bit 8) of IC_DATA_CMD register. Role of Realtek I2C: Slave-Transmitter
14	R	0	ABRT_SLV_ARBLOST	1: Slave lost the bus while transmitting data to a remote master. IC_TX_ABRT_SOURCE[12] is set at the same time. NOTE: Even though the slave never “owns” the bus, something could go wrong on the bus. This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then DW_apb_I2C no longer own the bus. Role of Realtek I2C: Slave-Transmitter
13	R	0	ABRT_SLVFLUSH_TXFIFO	1: Slave has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO. Role of Realtek I2C: Slave-Transmitter
12	R	0	ARB_LOST	1: Master has lost arbitration, or if IC_TX_ABRT_SOURCE[14] is also set, then the slave transmitter has lost arbitration. Role of Realtek I2C: Master-Transmitter or Slave-Transmitter
11	R	0	ABRT_MASTER_DIS	1: User tries to initiate a Master operation with the Master mode disabled.

				Role of Realtek I2C: Master-Transmitter or Master-Receiver
10	R	0	ABRT_10B_RD_NORSTRT	1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) = 0) and the master sends a read command in 10-bit addressing mode. Role of Realtek I2C: Master-Receiver
9	R	0	ABRT_SBYTE_NORSTRT	To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; restart must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (IC_TAR[11]), or the GC_OR_START bit must be cleared (IC_TAR[10]). Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, bit 9 clears for one cycle and then gets re-asserted. 1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) = 0) and the user is trying to send a START Byte. Role of Realtek I2C: Master
8	R	0	ABRT_HS_NORSTRT	1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) = 0) and the user is trying to use the master to transfer data in High Speed mode. Role of Realtek I2C: Master-Transmitter or Master-Receiver
7	R	0	ABRT_SBYTE_ACKDET	1: Master has sent a START Byte and the START Byte was acknowledged (wrong behavior). Role of Realtek I2C: Master
6	R	0	ABRT_HS_ACKDET	1: Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior). Role of Realtek I2C: Master
5	R	0	ABRT_GCALL_READ	1: DW_apb_I2C in master mode sent a General Call but the user programmed the byte following the General Call to be a read from the bus (IC_DATA_CMD[9] is set to 1). Role of Realtek I2C: Master-Transmitter
4	R	0	ABRT_GCALL_NOACK	1: DW_apb_I2C in master mode sent a General Call and no slave on the bus acknowledged the General Call. Role of Realtek I2C: Master-Transmitter
3	R	0	ABRT_TXDATA_NOACK	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). Role of Realtek I2C: Master-Transmitter
2	R	0	ABRT_10ADDR2_NOACK	1: Master is in 10-bit address mode and the second address byte of the 10-bit address was not acknowledged by any slave. Role of Realtek I2C: Master-Transmitter or Master-Receiver
1	R	0	ABRT_10ADDR1_NOACK	1: Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave. Role of Realtek I2C: Master-Transmitter or Master-Receiver

0	R	0	ABRT_7B_ADDR_NOACK	1: Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave. Role of Realtek I2C: Master-Transmitter or Master-Receiver
---	---	---	--------------------	---

12.3.34 IC_SLV_DATA_NACK_ONLY

- Name: I2C Generate Slave Data NACK Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-34 (Offset 0084h) IC_SLV_DATA_NACK_ONLY

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	RSVD.
0	R/W	0	IC_SLV_DATA_NACK_ONLY	Generate NACK. This NACK generation only occurs when I2C Module 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.

12.3.35 IC_DMA_CR

- Name: I2C DMA Control Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-35 (Offset 0088h) IC_DMA_CR

Bit	Access	INI	Symbol	Description
31:2	R	0	RSVD	RSVD.
1	R/W	0	IC_DMA_CR_TDMAE	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel.
0	R/W	0	IC_DMA_CR_RDMAE	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel.

12.3.36 IC_DMA_TDLR

- Name: I2C Transmit DMA Data Level Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-36 (Offset 008Ch) IC_DMA_TDLR

Bit	Access	INI	Symbol	Description
31:5	R	0	RSVD	RSVD.
4:0	R/W	0	IC_DMA_TDLR_DMATDL	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.

12.3.37 IC_DMA_RDLR

- Name: I2C Receive DMA Data Level Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-37 (Offset 0090h) IC_DMA_RDLR

Bit	Access	INI	Symbol	Description
31:5	R	0	RSVD	RSVD.
4:0	R/W	0	IC_DMA_RDLR_DMARDL	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or more than this field value, and RDMAE =1.

12.3.38 IC_SDA_SETUP

- Name: I2C Receive DMA Data Level Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-38 (Offset 0094h) IC_SDA_SETUP

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	RSVD.
7:0	R/W	0x05	IC_SDA_SETUP	SDA Setup. It is recommended that if the required delay is 1000ns, then for a sclk frequency of 10 MHz, REG_DW_I2C_IC_SDA_SETUP should be programmed to a value of 11. REG_DW_I2C_IC_SDA_SETUP must be programmed with a minimum value of 2.

12.3.39 IC_ACK_GENERAL_CALL

- Name: I2C ACK General Call Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-39 (Offset 0098h) IC_ACK_GENERAL_CALL

Bit	Access	INI	Symbol	Description
31:1	R	0	RSVD	RSVD.
0	R/W	1	IC_ACK_GENERAL_CALL	ACK General Call. When set to 1, I2C Module responds with a ACK (by asserting ic_data_oe) when it receives a General Call. When set to 0, the I2C Module does not generate General Call interrupts.

12.3.40 IC_ENABLE_STATUS

- Name: I2C Enable Status Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-40 (Offset 009Ch) IC_ENABLE_STATUS

Bit	Access	INI	Symbol	Description
31:5	R	0	RSVD	RSVD.
4:3	R	0	IC_ENABLE_STATUS_DMA_DISABLE_STATUS	DMA_DISABLE WHILE_BUSY.
2	R	0	IC_ENABLE_STATUS_SLV_RX_DATA_LOST	Slave Received Data Lost. This bit indicates if a Slave-Receiver operation has been aborted with at least one data byte received from an I2C transfer due to the setting of I2C_IC_ENABLE from 1 to 0.
1	R	0	IC_ENABLE_STATUS_SLV_DISABLED_LED WHILE_BUSY	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.
0	R	0	IC_ENABLE_STATUS_IC_EN	ic_en Status. This bit always reflects the value driven on the output port ic_en.

12.3.41 IC_DMA_CMD

- Name: I2C DMA Command Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-41 (Offset 00A0h) IC_DMA_CMD

Bit	Access	INI	Symbol	Description
31:8	R	0	RSVD	RSVD.
7	R/W	0	IC_DMA_CMD_RESTART	This bit controls whether a RESTART is issued before the next byte is sent or received.
6	R/W	0	IC_DMA_CMD_STOP	This bit controls whether a STOP is issued after the byte is sent or received.
5	R/W	0	IC_DMA_CMD_RW_CMD	This bit controls whether a read or a write is performed. This bit does not control the direction when the I2C Module acts as a slave. It controls only the direction when it acts as a master.
4:1	R/W	0	--	RSVD
0	R/W	0	IC_DMA_CMD_DMA_MODE_EN	Set to enable DMA mode.

12.3.42 IC_DMA_LEN

- Name: I2C DMA Length Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-42 (Offset 00A4h) IC_DMA_LEN

Bit	Access	INI	Symbol	Description
31:16	R	0	IC_DMA_TRD_LEN	DMA mode transfer bytes(Read only)
15:0	R/W	0	IC_DMA_LEN	DMA transfer data length(R/W)

12.3.43 IC_DMA_MOD

- Name: I2C DMA Mode Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-43 (Offset 00A8h) IC_DMA_MOD

Bit	Access	INI	Symbol	Description
31:2	R	0	--	RSVD
1:0	R/W	0	IC_DMA_CMD_DMA_MOD	When IC_DMA_CMD_DMA_MOD is equal to

12.3.44 IC_SLEEP

- Name: I2C SLEEP Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-44 (Offset 00ACh) IC_SLEEP

Bit	Access	INI	Symbol	Description
31:2	R	0	--	RSVD
1	R	0	IC_SLP_CLK_GATED	I2C clock has been gated (Read Only)
0	R/W	0	IC_CLK_CTRL	I2C clock control, write 1 controller would gate I2C clock until I2C slave is enable and reset synchronized register procedure is done.

12.3.45 IC_DEBUG_SEL

- Name: I2C Debug Mode Select Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-45 (Offset 00B0h) IC_DEBUG_SEL

Bit	Access	INI	Symbol	Description
31:4	R	0	--	RSVD
3:0	R/W	0	IC_DEBUG_SEL	Select the output signal of deport for I2C module

12.3.46 IC_STRCH_DLY

- Name: I2C Clock Stretch Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-46 (Offset 00B4h) IC_STRCH_DLY

Bit	Access	INI	Symbol	Description
31:3	R	0	--	RSVD
2:0	R/W	2	IC_OUT_SMP_DLY	Postponed determine the clock stretch timing.

12.3.47 IC_ANA_FLTR_SDA_RL

- Name: I2C Analog Filter SDA RL Select Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-47 (Offset 00B8h) IC_ANA_FLTR_SDA_RL

Bit	Access	INI	Symbol	Description
31:20	R	0	--	RSVD
19:0	R/W	0	IC_ANA_FLTR_SDA_RL	Determine the analog filter RL value of SDA

12.3.48 IC_ANA_FLTR_SDA_RM

- Name: I2C Analog Filter SDA RM Select Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-48 (Offset 00BCh) IC_ANA_FLTR_SDA_RM

Bit	Access	INI	Symbol	Description
31:20	R	0	--	RSVD
19:0	R/W	0	IC_ANA_FLTR_SDA_RM	Determine the analog filter RM value of SDA

12.3.49 IC_ANA_FLTR_SCL_RL

- Name: I2C Analog Filter SCL RL Select Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-49 (Offset 00C0h) IC_ANA_FLTR_SCL_RL

Bit	Access	INI	Symbol	Description
31:20	R	0	--	RSVD
19:0	R/W	0	IC_ANA_FLTR_SCL_RL	Determine the analog filter RL value of SCL

12.3.50 IC_ANA_FLTR_SCL_RM

- Name: I2C Analog Filter SCL RM Select Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-50 (Offset 00C4h) IC_ANA_FLTR_SCL_RM

Bit	Access	INI	Symbol	Description
31:20	R	0	--	RSVD
19:0	R/W	0	IC_ANA_FLTR_SCL_RM	Determine the analog filter RM value of SCL

12.3.51 IC_ANA_FLTR_SDA_CL

- Name: I2C Analog Filter SDA CL Select Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-51 (Offset 00C8h) IC_ANA_FLTR_SDA_CL

Bit	Access	INI	Symbol	Description
31:25	R	0	--	RSVD
24:0	R/W	0	IC_ANA_FLTR_SDA_CL	Determine the analog filter CL value of SDA

12.3.52 IC_ANA_FLTR_SDA_CM

- Name: I2C Analog Filter SDA CM Select Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-52 (Offset 00CCh) IC_ANA_FLTR_SDA_CM

Bit	Access	INI	Symbol	Description
31:25	R	0	--	RSVD
24:0	R/W	0	IC_ANA_FLTR_SDA_CM	Determine the analog filter CM value of SDA

12.3.53 IC_ANA_FLTR_SCL_CL

- Name: I2C Analog Filter SCL CL Select Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-53 (Offset 00D0h) IC_ANA_FLTR_SCL_CL

Bit	Access	INI	Symbol	Description
31:25	R	0	--	RSVD
24:0	R/W	0	IC_ANA_FLTR_SCL_CL	Determine the analog filter CL value of SCL

12.3.54 IC_ANA_FLTR_SCL_CM

- Name: I2C Analog Filter SCL CM Select Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-54 (Offset 00D4h) IC_ANA_FLTR_SCL_CM

Bit	Access	INI	Symbol	Description
31:25	R	0	--	RSVD
24:0	R/W	0	IC_ANA_FLTR_SCL_CM	Determine the analog filter CM value of SCL

12.3.55 IC_CLR_DMA_I2C_DONE

- Name: I2C Clear DMA I2C Done Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-55 (Offset 00E8h) IC_CLR_DMA_I2C_DONE

Bit	Access	INI	Symbol	Description
31:1	R	0	--	RSVD
0	R	0	IC_CLR_DMA_I2C_DONE	dma_I2C_done_intr interrupts raw status, and read clear

12.3.56 IC_FILTER

- Name: I2C FILTER Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-56 (Offset 00ECh) IC_FILTER

Bit	Access	INI	Symbol	Description
31:10	R	0	--	RSVD
9	R	0	--	RSVD
8	R/W	0	IC_DIG_FLTR_SEL	SCL/SDA spike filter setting, DIG_FLTR_SEL is to enable RX data filter. 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. When using address match without ic_clk, DIG_FLTR_SEL must be set to 0 for disabling digital filter.
7:4	R	0	--	RSVD
3:0	R/W	0	IC_DIG_FLTR_DEG	Digital filter degree

12.3.57 IC_SAR1

- Name: I2C Slave Address1 Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-57 (Offset 00F4h) IC_SAR1

Bit	Access	INI	Symbol	Description
31:7	R	0	--	RSVD
6:0	R/W	0x12	IC_SAR1	Another I2C slave address register since this I2C module supports dual slave addresses. When operating as I2C slave, this module could response REG_IC_SAR and REG_IC_SAR1 address from I2C master.

12.3.58 IC_COMP_VER

- Name: I2C Component Version Register
- Width: 32bit
- Initial Value: 0x0000

REG 12-58 (Offset 00FCh) IC_COMP_VER

Bit	Access	INI	Symbol	Description
31:0	R	0x2017 1208	IC_COMP_VER	0x20171208

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

13 Realtek DMA Controller (DMAC)

13.1 Overview

13.1.1 Application Scenario

Realtek Direct Memory Access Controller (RTK-DMAC) is a DMA controller with AXI interface. Data access is usually handled by CPU. When data is moved from source address to destination address, CPU sends sequential read/write commands controlling data access. However, CPU cannot execute instructions when it is handling the transfer. To release CPU resource, DMAC can manage the data transfer completely after CPU configures DMAC registers to setup a transfer. CPU does not have to take care of the transfer until DMAC trigger interrupts. The DMAC interrupts are generated when the transfer is done or the transfer encounters errors. When system loading is heavy, CPU resources are precious. With the help of DMA controller, we can largely save CPU resources and even to improve overall system performance.

13.1.2 Features

- 32 bits data bus width
- Two channels. Each channel can configure independent source address and destination address to initiate a transfer. The channel has a proprietary FIFO to push or pop data.
- The maximum transfer length per transfer is up to 4095 data items. Each data item can be configured to 1 byte, 2 bytes or 4 bytes width.
- DMA hardware request interface
 - Handshake with peripherals to control data flow
- Transfer abort feature
 - The transfer can be stopped safely. DMAC reports the correct data length already received or transmitted after the termination.

13.1.3 Architecture

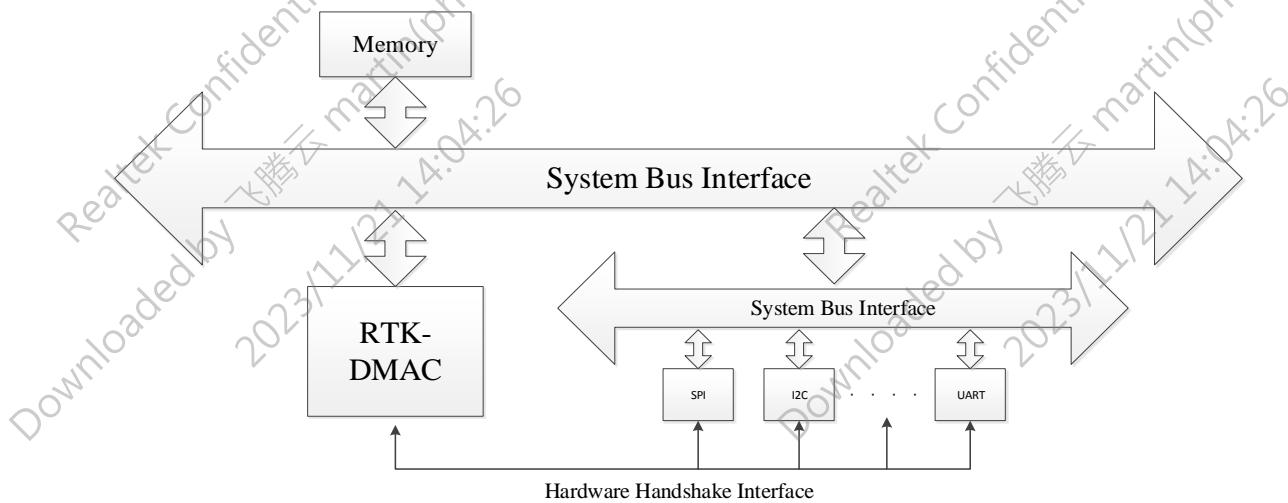


Figure 13-1 RTK-DMAC in A complete system

RTK-DMAC is an AXI bus master. It operates the same clock domain as AXI bus. RTK-DMAC handles data transfer between a source peripheral and a destination peripheral after the processor setup a transfer for it. The source peripheral or destination peripheral could be a memory device or a non-memory peripheral like SPI, UART or I2C. If the peripheral is a non-memory device, RTK-DMAC implements hardware handshake interfaces to manage the data transfer between source device and destination device. Each non-memory peripheral has propriety handshake signals and they are independent from other peripherals.

13.1.3.1 Functional Block Diagram

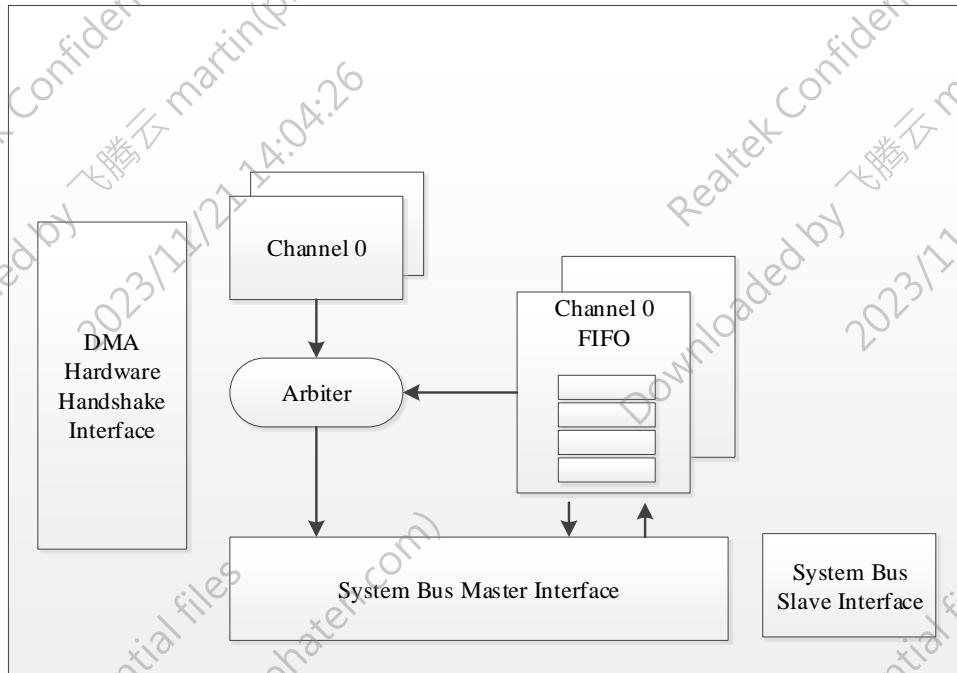


Figure 13-2 RTK-DMAC Block Diagram

- **System Bus Master Interface:** The AXI protocol master interface to achieve high-performance communication between source and destination peripherals. The master interfaces read data from a source peripheral and write it to a destination peripheral.
- **Slave Bus Interface:** A bus interface for low bandwidth control accesses. The processor configures RTK-DMAC register settings through this APB interface.
- **Arbiter:** When multiple channels are enabled simultaneously, arbiter determines which channel is permitted to transfer while the other channel holds the transfer according to the channel priority setting. The default priority is that channel 0 has higher priority than channel 1. Inside a channel, the source peripheral transfer has higher priority than the destination peripheral transfer.
- **Channel:** Read / write data path between a source peripheral and a destination peripheral.
- **Channel FIFO:** Each channel has independent channel FIFO. The FIFO depth is 16 bytes. This size is also the maximum burst transaction length. RTK-DMAC reads data from a source peripheral to the channel FIFO, then writes data out of the channel FIFO to a destination peripheral.
- **Hardware handshaking interface:** Uses hardware signals to control transferring a single or a burst transaction between the RTK-DMAC and the source or destination peripheral.

13.2 Functional Description

DMA Controller transfer data from a source device to a destination device over the AXI bus. DMA relevant registers are configured by CPU through APB bus, and then CPU initiates the transfer by enabling the DMA controller. After the transfer begins, the processor does not have to take care of it until an interrupt is triggered to notify CPU the transfer is complete or fails. CPU then jumps to an interrupt service routine to take over.

Memory to memory transfer is relatively simple. DMA Controller continuously transfers data because data is never overflowed on memory. This is not in case when the source address or destination address is non-memory peripheral, like SPI, UART or I2C. Pushing data length exceeding FIFO depth will lead to FIFO overflow. On the other hands, reading data from the FIFO when it is empty causes FIFO underflow. Both of these two errors should be prevented because data transfer may be abnormal. The DMA hardware handshake interface resolves these issues with five control signals. These control signals handshake with each other to control the data flow.

13.2.1 Overview

RTK-DMAC has two channels. Each channel can initiate a transfer separately. A channel has its proprietary FIFO to transfer data. Data is moved from a source peripheral to the channel FIFO, and then data is moved from the channel FIFO to a destination peripheral. By inducing this FIFO, the width of a data item and the burst transaction size can set different values for a source peripheral and a destination peripheral. The configuration of data width provides a flexibility to communicate with peripherals with various data width. The burst transaction size determines the number of data items being transferred per burst transaction between the FIFO and peripherals. This value is associated with channel FIFO depth, peripheral FIFO depth and data width. Performance can be improved by increasing bus utilization with optimal burst transaction size.

Each channel has two request lines that request ownership of the master bus interface: channel source and channel destination request lines. Several channels can be enabled simultaneously as long as their addresses do not overlap. However, these channels share the same AXI bus to transfer data. The arbiter is employed to control data transfer. 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 AXI bus, then AXI transfers can begin between the peripheral and DMAC. The arbiter decides which

of the request lines, $2 * \text{Total channel number}$, is granted the master bus interface. The lower channel number has the higher priority. Within a channel, a channel source is granted before the destination if both have their request lines asserted when a grant decision is made. High priority request cannot preempt the transfer proceeded by a request with lower priority. To prevent a channel from saturating the master interface, a request line is allowed to transfer data that the maximum data length is the maximum AXI burst length. The maximum AXI burst length is a function of the channel FIFO depth and the bus data width. After the request line is served, the arbiter switches the grant line to next asserted request line with the highest priority.

DMA hardware handshake interfaces are used to communicate with non-memory peripherals to prevent peripheral FIFO underflow errors or peripheral FIFO overflow errors. More details are illustrated on the next section. The DMA registers are programmed through APB3 interface. Register access, either read or write, is always word access.

13.2.2 DMA Hardware Handshake

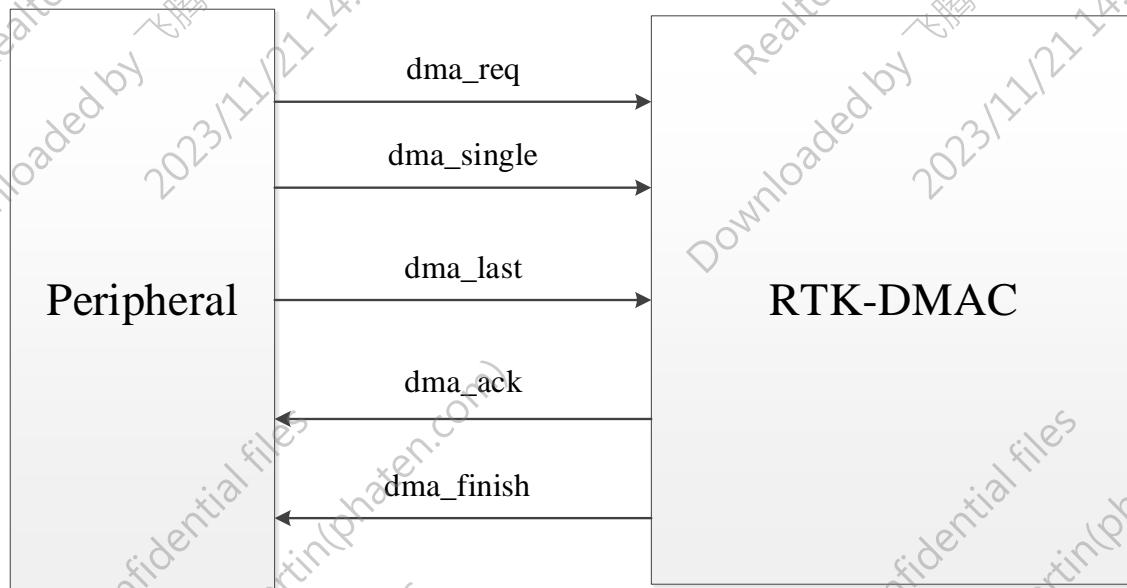


Figure 13-3 DMA Hardware Handshake Signals

DMA controller controls data flow of non-memory peripherals through the hardware handshake interface. The five signals determine when to transmit data to a peripheral's FIFO and when to receive data from the FIFO. Dma_req, dma_single and dma_last are input signals

asserted by peripherals while dma_ack and dma_finish are output signals of RTK-DMAC.

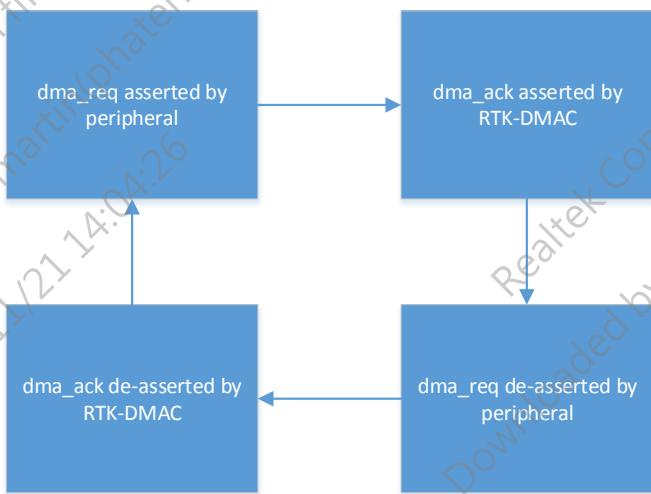


Figure 13-4 DMA Handshaking Loop

Non-memory peripheral has several registers configuring for DMA transfer. These registers set up threshold values when the peripheral communicates with DMA. Once the threshold value is reached, the peripheral sends a request signal to the DMA by asserting dma_req signal. After DMA is aware that a request is sent, it responds with a burst transaction before asserting dma_ack signal. The burst transaction will send a block of data which the number of data items is configured in CTLx. dma_req then de-asserts when it detects dma_ack is asserted. dma_ack is de-asserted a half cycle later to finish the DMA handshake process. dma_finish indicates the transaction is the last transaction of the transfer. DMA asserts dma_finish to inform the peripheral that it has transferred all data. At the end of the transfer, dma_finish is asserted along with dma_ack and they are de-asserted at the same time. If the transfer length is not aligned to bus width, the peripheral asserts dma_single to transfer one data item per transaction instead of asserting dma_req. Since the DMA controller is always the data flow controller, dma_last, which is asserted by the peripheral to indicate the end of the transfer, is not used.

13.2.3 Peripheral Burst Transaction Requests

Transaction is the basic unit of a RTK-DMAC transfer. A transaction is relevant only for transfers between a non-memory peripheral and RTK-DMAC. There are two types of transactions. Single transaction is less efficient because the length of a transaction is always 1. It only carries one data item per transaction. On the other hands, burst transactions is converted into a sequence of bursts and AXI single transfers. The length of a burst transaction is

configurable by the CTLx register. The maximum data bytes transferred by a burst transaction (burst transaction size) is limited by the channel FIFO depth, 16 bytes.

The burst transaction size is governed by two parameters MSIZE and TR_WIDTH. Since data is moved from the source peripheral to the channel FIFO before data is written to the destination peripheral, these two parameters can be configured separately.

- SRC_MSIZE / DST_MSIZE: The value represents the number of data item in a burst transaction.
- SRC_TR_WIDTH / DST_TR_WIDTH: The width of a data item.

The burst transaction size = SRC_MSIZE * SRC_TR_WIDTH or DST_MSIZE * DST_TR_WIDTH
TR_WIDTH is associated with the non-memory peripheral. It depends on the width of the peripheral FIFO. MSIZE is determined by users but some constraints still need to be concerned. First, the burst transaction size should not exceed 16 bytes. Second, inappropriate burst transaction size may cause peripheral FIFO underflow or overflow.

As illustrated on the previous section, the dma_req signal is asserted by the non-memory peripheral when its FIFO reaches a threshold value. DMAC responds the request signal by a burst transaction size of data. If the empty space of peripheral FIFO is smaller than the burst transaction size, the peripheral FIFO may overflow. On the contrary, if the empty space is larger, it is possible the peripheral consumes data too fast that it asserts dma_req more frequently or even the FIFO is underflow. To improve bus utilization and prevent FIFO overflow / underflow, the threshold value should be considered in conjunction with the burst transaction size.

For example, consider a transfer from a memory device to a destination peripheral SPI. The SPI is configured to 8 bit mode with FIFO depth = 64 bytes. If we set the threshold value to 48, which means SPI asserts dma_req when the number of data item is equal or lower than 48, the optimal value of the burst transaction size should be 16. The SPI FIFO leaves 16 bytes space at the time the dma_req is asserted, DMAC responds by the burst transaction size of data 16 bytes to just fill the SPI FIFO. Therefore, the SRC_TR_WIDTH = 4 bytes (memory data width) and the SRC_MSIZE = 4.

13.2.4 RTK-DMAC Interrupts

DMAC supports five interrupt requests, each of which can be masked. Unless users have special requirement for their applications, only IntTft and IntErr are enabled. The other interrupt bits status still can be read from raw status registers.

13.2.4.1 Block Transfer Complete Interrupt (IntBlock)

The interrupt is generated on DMA block transfer completion to the destination peripheral. This interrupt signal is asserted with DMA Transfer Complete Interrupt.

13.2.4.2 Destination Transaction Complete Interrupt (IntDstTran)

The interrupt is triggered after completion of the last transfer of the requested transaction from the handshaking interface on the destination side.

13.2.4.3 Source Transaction Complete Interrupt (IntSrcTran)

The interrupt is triggered after completion of the last transfer of the requested transaction from the handshaking interface on the source side.

13.2.4.4 Error Interrupt (IntErr)

This interrupt is generated when an error response is received from AXI bus during transfer.

13.2.4.5 DMA Transfer Complete Interrupt (IntTfr)

The interrupt is generated when DMA transfer is complete. This interrupt happens along with IntDstTran, IntSrcTran and IntBlock.

13.3 Control Registers

13.3.1 DMA_SARx

- Name: Source Address Register for Channel x
- Width: 64 bits
- Address Offset:
 - SAR0 – 0x000
 - SAR1 – 0x058
- Initial Value: 0x0000

The starting source address of a DMA transfer is programmed through the SARx register. It should be programmed by the processor before the DMA channel is enabled. The address can be unaligned to the width of a data item, CTLx.SRC_TR_WIDTH.

REG 13-1 DMA_SARx

Bits	Access	INI	Symbol	Description
63:0	R/W	0x0	SAR	Source Address of DMA transfer.

13.3.2 DMA_DARx

- Name: Destination Address Register for Channel x
- Width: 64 bits
- Address Offset:
 - DAR0 – 0x008
 - DAR1 – 0x060
- Initial Value: 0x0000

The starting destination address of a DMA transfer is programmed through the DARx register. It should be programmed by the processor before the DMA channel is enabled. The address can be unaligned to the width of a data item, CTLx.DST_TR_WIDTH.

REG 13-2 DMA_DARx

Bits	Access	INI	Symbol	Description
63:0	R/W	0x0	DAR	Destination Address of DMA transfer.

13.3.3 DMA_CTLx

- Name: Control Register for Channel x
- Width: 64 bits
- Address Offset:
 - CTL0 – 0x018
 - CTL1 – 0x070
- Initial Value: 0x200004825

This register contains fields that control the DMA transfer. This register must be programmed prior to enabling the channel.

REG 13-3 DMA_CTLx

Bits	Access	INI	Symbol	Description
43:32	R/W	0x2	BLOCK_TS	Block Transfer Size. When the RTK_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 AXI beat. The width of the single transaction is determined by CTLx.SRC_TR_WIDTH.
22:20	R/W	0x0	TT_FC	Transfer Type and Flow Control. It indicates the type of the source peripheral and the destination peripheral. If the target device is non-memory peripheral, handshake interface is necessary. Memory to Memory Memory to Peripheral Peripheral to Memory
16:14	R/W	0x1	SRC_MSIZEx	Source Burst Transaction Length. It indicates the number of data items to be read from source for a burst transaction. When a source peripheral sends a burst transaction request, DMAC reads that amount of data items from the peripheral.
13:11	R/W	0x1	DEST_MSIZEx	Destination Burst Transaction Length. It indicates the number of data items to be written to destination for a burst transaction. When a destination peripheral sends a burst transaction request, DMAC writes that amount of data items to the peripheral.
10:9	R/W	0x0	SINC	Source Address Increment. Indicates whether to increment the source address on every source transfer or keep the source address with the same value (for FIFO). The increment of each source transfer is aligned to SRC_TR_WIDTH. 2'b00 = Increment 2'b1x = No channel
8:7	R/W	0x0	DINC	Destination Address Increment. Indicates whether to increment the destination address on every destination transfer or keep the destination address with the same value (for FIFO). The increment of each destination transfer is aligned to DST_TR_WIDTH. 2'b00 = Increment 2'b1x = No channel

6:4	R/W	0x10	SRC_TR_WIDTH	<p>Source Transfer Width. The width of a data item for source peripheral. This value should not exceed the bus width, 32 bits. This bit field is also used to calculate total block data size, i.e. the number of data items. The number of data items is derived from the formula:</p> $\text{Number of data item} = \frac{\text{Total transfer bytes}}{\text{SRC_TD_WIDTH}}$ <p>The decoding values for different widths are shown in Table 3.2.</p>
3:1	R/W	0x10	DST_TR_WIDTH	Destination Transfer Width. The width of a data item for destination peripheral. If the destination peripheral is not memory, this value is typically the same as the destination peripheral FIFO width. The decoding values for different widths are shown in Table 3.2.
0	R/W	0x1	INT_EN	Interrupt Enable Bit. Set the bit to enable all interrupt-generating sources. It is a global mask bit for all interrupts of the channel. Raw interrupt bits are still valid even if INT_EN = 0.

Table 13-4 CTLx.SRC_MSIZE/DEST_MSIZE Decoding

SRC_MSIZE/DEST_MSIZE	Number of data items to be transferred (width is defined in SRC_TR_WIDTH and DST_TR_WIDTH)
000	1
001	4
010	8
011	16

Table 13-5 CTLx.SRC_TR_WIDTH/DEST_TR_WIDTH Decoding

SRC_TR_WIDTH/DEST_TR_WIDTH	Size (bits)
000	8
001	16
010	32

Table 13-6 CTLx.TT_FC Field Decoding

TT_FC	Transfer Type	Flow Controller
000	Memory to Memory	DMAC
001	Memory to Peripheral	DMAC
010	Peripheral to Memory	DMAC

13.3.4 DMA_CFGx

- Name: Configuration Register for Channel x
- Width: 64 bits
- Address Offset:
CFG0 – 0x040

CFG1 – 0x098

- Initial Value: 0x800000E00

This register contains fields that configure the DMA transfer.

REG 13-7 DMA_CFGx

Bits	Access	INI	Symbol	Description
46:43	R/W	0x0	DEST_PER	Assigns a hardware handshaking interface number to the non-memory destination peripheral of the channel x. The number can be 0~31 for high speed platform and 0~7 for low speed platform. Either on high speed platform or low speed platform, each peripheral corresponds to a specified value which is determined by hardware internally. The channel communicates with the target peripheral connected to that interface through the assigned hardware handshaking interface. Only one peripheral should be assigned to the same handshaking interface.
42:39	R/W	0x0	SRC_PER	Assigns a hardware handshaking interface number to the non-memory source peripheral of the channel x. The number can be 0~31 for high speed platform and 0~7 for low speed platform. Either on high speed platform or low speed platform, each peripheral corresponds to a specified value which is determined by hardware internally. The channel communicates with the target peripheral connected to that interface through the assigned hardware handshaking interface. Only one peripheral should be assigned to the same handshaking interface.
35	R/W	0x1	RSVD	Reserved
11	R/W	0x1	HS_SEL_SRC	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. Only support this handshaking interface. 1 – Software handshaking interface. Not support. If the source peripheral is memory, then this bit is ignored.
10	R/W	0x1	HS_SEL_DST	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. Only support this handshaking interface. 1 – Software handshaking interface. Not support. If the destination peripheral is memory, then this bit is ignored.
9	R	0x1	FIFO_EMPTY	Indicate if there is data left in the channel FIFO. This bit can be used in conjunction with CH_SUSP to safely disable a channel. 0 – Channel FIFO not empty 1 – Channel FIFO empty
8	R/W	0x0	CH_SUSP	Channel Suspend. Suspend all DMA data transfers from the source until this bit is cleared. FIFO_EMPTY bit can be used to check FIFO status after channel suspension. DMA may not stop immediately when this bit is set. After CH_SUSP is set, the INACTIVE, SRC_PCTL_OVER and DST_PCTL_OVER should be read to ensure current transaction is complete before disabling the channel to terminate the transfer. This provides a safe way to stop the transfer without losing data. Once the transfer is

				suspended, it cannot be resumed. The CH_SUSP must be clear to initiate a new transfer. 0 – Not suspended 1 – Suspend DMA transfer from the source
2	R	0x0	DST_PCTL_OVER	DMA destination bus protocol finishes. Indicate the current write command is done.
1	R	0x0	SRC_PCTL_OVER	DMA source bus protocol finishes. Indicate the current read command is done.
0	R	0x0	INACTIVE	Indicate if the channel is inactive. It can be used in conjunction with CH_SUSP to cleanly disable a channel when destination is a non-memory peripheral. 1 – Channel inactive 0 – Channel not inactive

13.3.5 DMA_RawTfr

- Name: DMA Transfer Complete Interrupt Raw Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows the raw interrupt status of DMA Transfer Complete Interrupt whether or not the interrupt is masked. Each bit corresponds to one channel.

REG 13-8 (Offset 02C0h) DMA_RawTfr

Bits	Access	INI	Symbol	Description
1:0	R	0x0	RAW_TFR	Raw interrupt status of DMA Transfer Complete Interrupt.

13.3.6 DMA_RawBlock

- Name: Block Transfer Complete Interrupt Raw Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows the raw interrupt status of Block Transfer Complete Interrupt whether or not the interrupt is masked. Each bit corresponds to one channel.

REG 13-9 (Offset 02C8h) DMA_RawBlock

Bits	Access	INI	Symbol	Description
1:0	R	0x0	RAW_BLOCK	Raw interrupt status of Block Transfer Complete Interrupt.

13.3.7 DMA_RawSrcTran

- Name: Source Transaction Complete Interrupt Raw Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows the raw interrupt status of Source Transaction Complete Interrupt whether or not the interrupt is masked. Each bit corresponds to one channel.

REG 13-10 (Offset 02D0) DMA_RawSrcTran

Bits	Access	INI	Symbol	Description
1:0	R	0x0	RAW_SRC_TRAN	Raw interrupt status of Source Transaction Complete Interrupt.

13.3.8 DMA_RawDstTran

- Name: Destination Transaction Complete Interrupt Raw Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows the raw interrupt status of Destination Transaction Complete Interrupt whether or not the interrupt is masked. Each bit corresponds to one channel.

REG 13-11 (Offset 02D8h) DMA_RawDstTran

Bits	Access	INI	Symbol	Description
1:0	R	0x0	RAW_DST_TRAN	Raw interrupt status of Destination Transaction Complete Interrupt.

13.3.9 DMA_RawErr

- Name: Error Interrupt Raw Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows the raw interrupt status of Error Interrupt whether or not the interrupt is masked. Each bit corresponds to one channel.

REG 13-12 (Offset 02E0h) DMA_RawErr

Bits	Access	INI	Symbol	Description
1:0	R	0x0	RAW_ERR	Raw interrupt status of Error Interrupt.

13.3.10 DMA_StatusTfr

- Name: DMA Transfer Complete Interrupt Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows the interrupt status of DMA Transfer Complete Interrupt. Each bit corresponds to one channel.

REG 13-13 (Offset 02E8h) DMA_StatusTfr

Bits	Access	INI	Symbol	Description
1:0	R	0x0	STATUS_TFR	Interrupt status of DMA Transfer Complete Interrupt.

13.3.11 DMA_StatusBlock

- Name: Block Transfer Complete Interrupt Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows the interrupt status of Block Transfer Complete Interrupt. Each bit corresponds to one channel.

REG 13-14 (Offset 02F0h) DMA_StatusBlock

Bits	Access	INI	Symbol	Description
1:0	R	0x0	STATUS_BLOCK	Interrupt status of Block Transfer Complete Interrupt.

13.3.12 DMA_StatusSrcTran

- Name: Source Transaction Complete Interrupt Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows the interrupt status of Source Transaction Complete Interrupt. Each bit

corresponds to one channel.

REG 13-15 (Offset 02F8h) DMA_StatusSrcTran

Bits	Access	INI	Symbol	Description
1:0	R	0x0	STATUS_SRC_TRA_N	Interrupt status of Source Transaction Complete Interrupt.

13.3.13 DMA_StatusDstTran

- Name: Destination Transaction Complete Interrupt Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows the interrupt status of Destination Transaction Complete Interrupt. Each bit corresponds to one channel.

REG 13-16 (Offset 0300h) DMA_StatusDstTran

Bits	Access	INI	Symbol	Description
1:0	R	0x0	STATUS_DST_TRA_N	Interrupt status of Destination Transaction Complete Interrupt.

13.3.14 DMA_StatusErr

- Name: Error Interrupt Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows the interrupt status of Error Interrupt. Each bit corresponds to one channel.

REG 13-17 (Offset 0308h) DMA_StatusErr

Bits	Access	INI	Symbol	Description
1:0	R	0x0	STATUS_ERR	Interrupt status of Error Interrupt.

13.3.15 DMA_MaskTfr

- Name: DMA Transfer Complete Interrupt Mask Register
- Width: 64 bits

■ Initial Value: 0x0000

The register masks the content of raw status of DMA Transfer Complete Interrupt. If a particular channel of the interrupt is unmasked, the processor can receive the interrupt signal when the corresponding bit of that channel on the raw status register is set. To configure the mask of a channel, the INR_MASK_TFR_WE, interrupt mask write enable bit, should be programmed with the INT_TFR_MASK for that channel. For example, if the channel x unmasks the interrupt, the bit x and the bit $(8 + x)$ should be programmed at the same time. Each bit corresponds to one channel.

REG 13-18 (Offset 0310h) DMA_MaskTfr

Bits	Access	INI	Symbol	Description
9:8	W	0x0	INT_TFR_MASK_WE	Interrupt Mask Write Enable of DMA Transfer Complete Interrupt. 0 – write disabled 1 – write enabled
1:0	R/W	0x0	INT_TFR_MASK	Interrupt Mask of DMA Transfer Complete Interrupt. 0 – masked 1 – unmasked (enabled)

13.3.16 DMA_MaskBlock

- Name: Block Transfer Complete Interrupt Mask Register
- Width: 64 bits
- Initial Value: 0x0000

The register masks the content of raw status of Block Transfer Complete Interrupt. If a particular channel of the interrupt is unmasked, the processor can receive the interrupt signal when the corresponding bit of that channel on the raw status register is set. To configure the mask of a channel, the INR_MASK_BLOCK_WE, interrupt mask write enable bit, should be programmed with the INT_BLOCK_MASK for that channel. For example, if the channel x unmasks the interrupt, the bit x and the bit $(8 + x)$ should be programmed at the same time. Each bit corresponds to one channel.

REG 13-19 (Offset 0318h) DMA_MaskBlock

Bits	Access	INI	Symbol	Description
9:8	W	0x0	INT_BLOCK_MASK_WE	Interrupt Mask Write Enable of Block Transfer Complete Interrupt. 0 – write disabled 1 – write enabled

1:0	R/W	0x0	INT_BLOCK_MASK	Interrupt Mask of Block Transfer Complete Interrupt. 0 – masked 1 – unmasked (enabled)
-----	-----	-----	----------------	--

13.3.17 DMA_MaskSrcTran

- Name: Source Transaction Complete Interrupt Mask Register
- Width: 64 bits
- Initial Value: 0x0000

The register masks the content of raw status of Source Transaction Complete Interrupt. If a particular channel of the interrupt is unmasked, the processor can receive the interrupt signal when the corresponding bit of that channel on the raw status register is set. To configure the mask of a channel, the INR_MASK_SRC_TRAN_WE, interrupt mask write enable bit, should be programmed with the INT_SRC_TRAN_MASK for that channel. For example, if the channel x unmasks the interrupt, the bit x and the bit $(8 + x)$ should be programmed at the same time. Each bit corresponds to one channel.

REG 13-20 (Offset 0320h) DMA_MaskSrcTran

Bits	Access	INI	Symbol	Description
9:8	W	0x0	INT_SRC_TRAN_MASK_WE	Interrupt Mask Write Enable of Source Transaction Complete Interrupt. 0 – write disabled 1 – write enabled
1:0	R/W	0x0	INT_SRC_TRAN_MASK	Interrupt Mask of Source Transaction Complete Interrupt. 0 – masked 1 – unmasked (enabled)

13.3.18 DMA_MaskDstTran

- Name: Destination Transaction Complete Interrupt Mask Register
- Width: 64 bits
- Initial Value: 0x0000

The register masks the content of raw status of Destination Transaction Complete Interrupt. If a particular channel of the interrupt is unmasked, the processor can receive the interrupt signal when the corresponding bit of that channel on the raw status register is set. To configure the mask of a channel, the INR_MASK_DST_TRAN_WE, interrupt mask write enable bit, should be programmed with the INT_DST_TRAN_MASK for that channel. For example, if

the channel x unmasks the interrupt, the bit x and the bit $(8 + x)$ should be programmed at the same time. Each bit corresponds to one channel.

REG 13-21 (Offset 0328h) DMA_MaskDstTran

Bits	Access	INI	Symbol	Description
9:8	W	0x0	INT_DST_TRAN_ MASK _{WE}	Interrupt Mask Write Enable of Destination Transaction Complete Interrupt. 0 – write disabled 1 – write enabled
1:0	R/W	0x0	INT_DST_TRAN_ MASK	Interrupt Mask of Destination Transaction Complete Interrupt. 0 – masked 1 – unmasked (enabled)

13.3.19 DMA_MaskErr

- Name: Error Interrupt Mask Register
- Width: 64 bits
- Initial Value: 0x0000

The register masks the content of raw status of Error Interrupt. If a particular channel of the interrupt is unmasked, the processor can receive the interrupt signal when the corresponding bit of that channel on the raw status register is set. To configure the mask of a channel, the INR_MASK_ERR_WE, interrupt mask write enable bit, should be programmed with the INT_ERR_MASK for that channel. For example, if the channel x unmasks the interrupt, the bit x and the bit $(8 + x)$ should be programmed at the same time. Each bit corresponds to one channel.

REG 13-22 (Offset 0330h) DMA_MaskErr

Bits	Access	INI	Symbol	Description
9:8	W	0x0	INT_ERR_MASK _{WE}	Interrupt Mask Write Enable of Error Interrupt. 0 – write disabled 1 – write enabled
1:0	R/W	0x0	INT_ERR_MASK	Interrupt Mask of Error Interrupt. 0 – masked 1 – unmasked (enabled)

13.3.20 DMA_ClearTfr

- Name: DMA Transfer Complete Interrupt Clear Register
- Width: 64 bits
- Initial Value: 0x0000

The register clears DMA Transfer Complete Interrupt. The channel bit in DMA Transfer Complete Interrupt Raw Status Register and DMA Transfer Complete Interrupt Status Register are cleared on the same cycle when the corresponding bit of this register is writing to 1. Each bit corresponds to one channel.

REG 13-23 (Offset 0338h) DMA_ClearTfr

Bits	Access	INI	Symbol	Description
1:0	W	0x0	INT_TFR_CLEAR	Clear DMA Transfer Complete Interrupt. 0 – no effect 1 – clear interrupt

13.3.21 DMA_ClearBlock

- Name: Block Transfer Complete Interrupt Clear Register
- Width: 64 bits
- Initial Value: 0x0000

The register clears Block Transfer Complete Interrupt. The channel bit in Block Transfer Complete Interrupt Raw Status Register and Block Transfer Complete Interrupt Status Register are cleared on the same cycle when the corresponding bit of this register is writing to 1. Each bit corresponds to one channel.

REG 13-24 (Offset 0340h) DMA_ClearBlock

Bits	Access	INI	Symbol	Description
1:0	W	0x0	INT_BLOCK_CLEAR	Clear Block Transfer Complete Interrupt. 0 – no effect 1 – clear interrupt

13.3.22 DMA_ClearSrcTran

- Name: Source Transaction Complete Interrupt Clear Register
- Width: 64 bits
- Initial Value: 0x0000

The register clears Source Transaction Complete Interrupt. The channel bit in Source

Transaction Complete Interrupt Raw Status Register and Source Transaction Complete Interrupt Status Register are cleared on the same cycle when the corresponding bit of this register is writing to 1. Each bit corresponds to one channel.

REG 13-25 (Offset 0348h) DMA_ClearSrcTran

Bits	Access	INI	Symbol	Description
1:0	W	0x0	INT_SRC_TRAN_C LEAR	Clear Source Transaction Complete Interrupt. 0 – no effect 1 – clear interrupt

13.3.23 DMA_ClearDstTran

- Name: Destination Transaction Complete Interrupt Clear Register
- Width: 64 bits
- Initial Value: 0x0000

The register clears Destination Transaction Complete Interrupt. The channel bit in Destination Transaction Complete Interrupt Raw Status Register and Destination Transaction Complete Interrupt Status Register are cleared on the same cycle when the corresponding bit of this register is writing to 1. Each bit corresponds to one channel.

REG 13-26 (Offset 0350h) DMA_ClearDstTran

Bits	Access	INI	Symbol	Description
1:0	W	0x0	INT_DST_TRAN_C LEAR	Clear Destination Transaction Complete Interrupt. 0 – no effect 1 – clear interrupt

13.3.24 DMA_ClearErr

- Name: Error Interrupt Clear Register
- Width: 64 bits
- Initial Value: 0x0000

The register clears Error Interrupt. The channel bit in Error Interrupt Raw Status Register and Error Interrupt Status Register are cleared on the same cycle when the corresponding bit of this register is writing to 1. Each bit corresponds to one channel.

REG 13-27 (Offset 0358h) DMA_ClearErr

Bits	Access	INI	Symbol	Description
1:0	W	0x0	INT_ERR_CLEAR	Clear Error Interrupt. 0 – no effect 1 – clear interrupt

13.3.25 DMA_StatusInt

- Name: Combined Interrupt Status Register
- Width: 64 bits
- Initial Value: 0x0000

The register shows ORed contents of the five status registers – StatusTfr, StatusBlock, StatusSrcTran, StatusDstTran and StatusErr.

REG 13-28 (Offset 0360h) DMA_StatusInt

Bits	Access	INI	Symbol	Description
4	R	0x0	ERR	OR of the content of StatusErr register.
3	R	0x0	DSTT	OR of the content of StatusDstTran register.
2	R	0x0	SRCT	OR of the content of StatusSrcTran register.
1	R	0x0	BLOCK	OR of the content of StatusBlock register.
0	R	0x0	TFR	OR of the content of StatusTfr register.

13.3.26 DMA_DmaCfgReg

- Name: Configuration Register
- Width: 64 bits
- Initial Value: 0x0000

The register is used to enable DMAC, which must be done before any channel activity can begin.

REG 13-29 (Offset 0398h) DMA_DmaCfgReg

Bits	Access	INI	Symbol	Description
0	R/W	0x0	DMA_EN	DMAC Enable bit. 0 – DMAC is disabled. 1 – DMAC is enabled.

13.3.27 DMA_ChEnReg

- Name: Channel Enable Register
- Width: 64 bits
- Initial Value: 0x0000

The register is used to enable channels. If software needs to set up a new channel, it can read this register to find out which channels are currently inactive. Once the channel is enabled, the transfer begins immediately. Before enabling the channel bit, all relevant settings of the transfer, including SAR, DAR, CTL, CFG and Interrupt Mask, should be configured. The channel bit is automatically cleared by hardware after the transfer finishes.

To enable a channel, channel enable write enable bit, CH_EN_WE, should be programmed with CH_EN. For example, if users want to enable the channel x , the bit x and the bit $(8 + x)$ should be programmed at the same time. Each bit corresponds to one channel.

REG 13-30 (Offset 03A0h) DMA_ChEnReg

Bits	Access	INI	Symbol	Description
9:8	W	0x0	CH_EN_WE	Channel enable write enable.
1:0	R/W	0x0	CH_EN	Enable/Disable the channel. Set this bit to enable a channel and clear this bit to disable a channel. The transfer is terminated when the channel is disabled. However, DMAC cannot guarantee no data lost if the transfer is stopped before completion. Please follow standard transfer abort flow to stop the transfer instead of disabling the channel directly.

13.3.28 DMA_SoftResetReg

- Name: Soft Reset Register
- Width: 64 bits
- Initial Value: 0x0000

The register is used to reset channels. To reset a channel, channel reset write enable bit, CH_RESET_EN_WE, should be programmed with CH_RESET_EN. Two steps to reset a channel. If users want to reset the channel x , the bit x and the bit $(8 + x)$ should be programmed to 1 at the same time. After one cycle, program $(8 + x)$ to 1 and x to 0 to release the channel from reset state. The reset bit will not clear automatically by hardware. Each bit corresponds to one channel.

REG 13-31 (Offset 03B8h) DMA_SoftResetReg

Bits	Access	INI	Symbol	Description
9:8	W	0x0	CH_RESET_EN_W_E	Channel reset write enable.
1:0	R/W	0x0	CH_RESET_EN	Reset the channel. We set this bit to reset a channel while we clear this bit to finish the reset flow and release the channel from reset state. 1 – reset the channel 0 – release the channel from reset state

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

Realtek Confidential files
Downloaded by 飞腾云 martin(phaten.com)
2023/11/21 14:04:26

14 SDIO Device/SPI Slave Interface

14.1 Overview

14.1.1 Application Scenario

A system can have a host platform and an Ameba-Z II coexist on the same board. The SDIO interface is one of the selections for the communication between Host platform and Ameba-Z II. Compare to other type of interface, like UART or SPI, the SDIO interface can provides much higher data rate. So the SDIO interface normally is used for applications those need higher data rate for the communication between host CPU and Ameba-Z II. A common use scenario is the Ameba-Z II works as a SDIO Wi-Fi network interface card to provide the wireless LAN service for the host platform. Figure 14-1 shows the architecture of this type of system.

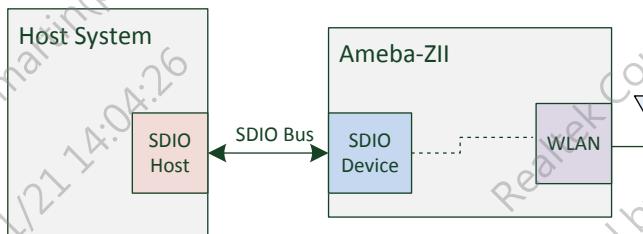


Figure 14-1 Ameba-Z II Works as An SDIO Device In A System

Ameba-Z II also supports a high speed SPI slave interface. However this SPI interface shares the same I/O pins and partial of hardware with the SDIO device interface. That means the SDIO device mode interface and the high speed SPI slave mode interface cannot work concurrently. The Ameba-Z II can be configured to work as either a SDIO device or a SPI slave device. This design provides more flexibility on the connective of Ameba-Z II with different type of host platform. Figure 14-2 shows an example of Ameba-Z II works as a SPI Slave device. There is an extra pin needed for this SPI Slave mode. This pin is used to send a signal to trigger an interrupt of the Host system. This implements the similar mechanism of SDIO Interrupt transfer. Ameba-Z II use this interrupt to notify the events of received packets ready, power status changing or error detection.

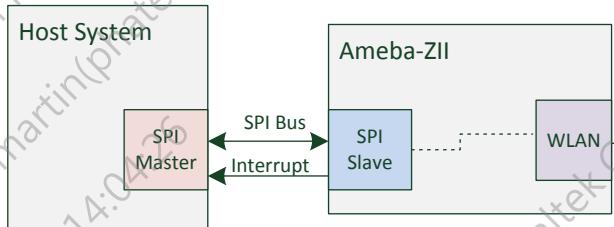


Figure 14-2 Ameba-Z II Works as An SPI Slave Device In A System

At the system initialization, the host needs to make the bus type configuration first. This can be achieved by a writing of the Ameba-Z II's SPI configuration register. Ameba-Z II will enable this SDIO/SPI interface as SPI mode default. If the 1st bus transfer is a SPI transfer with the writing of the SPI configuration register, the SPI mode will be kept. Otherwise Ameba-Z II hardware will switch this SDIO/SPI interface to the SDIO device mode automatically.

14.1.2 Features

■ SDIO device mode:

- Full compliance with SDIO card specification version 2.0.
- Supports data transfer in block(s) mode.
- Supported speed:

Table 14-1 SDIO Speed

Mode	Max. Data Rate	Max. SDIO CLK	I/O Voltage
Default Speed	12.5 MB/S	25 MHz	3.3V
High Speed	25 MB/S	50 MHz	3.3V

- Dedicated DMA engine for data transfer to reduce CPU loading.

■ SPI Slave mode

- Full compliance with general SPI bus specification.
- Support up to 25MHz SPI clock rate.
- Dedicated DMA engine for data transfer to reduce CPU loading.

14.1.3 Architecture

Figure 14-3 shows the hardware architecture of SDIO Device / SPI Slave interface. For power saving consideration, this hardware architecture is designed to be composed by 2 parts: On Domain and Off Domain. The On Domain is the part which is always powered on; the Off Domain will be turned off when Ameba-Z II is in the sleep mode. There 2 On Domains for SDIO device and the SPI Slave interface separately. The On Domain implements the interface bus protocol and mainly is controlled by the host side. The Off Domain mainly is controlled by the Ameba-Z II's CPU, which implement the data packets DMA transfer. There is one Off Domain which can work with either SDIO Device or SPI Slave interface only.

The SDIO Device and the SPI Slave also share the I/O pins. The pins sharing are list in the following table:

Table 14-2 SDIO/SPI Pin Table

Pin Name	SDIO Pins	SPI Pins
A15	SDIO D2	--
A16	SDIO D3	SPI CS
A17	SDIO CMD	MOSI
A18	SDIO CLK	SPI CLK
A19	SDIO D0	MISO
A20	SDIO D1	SPI Interrupt

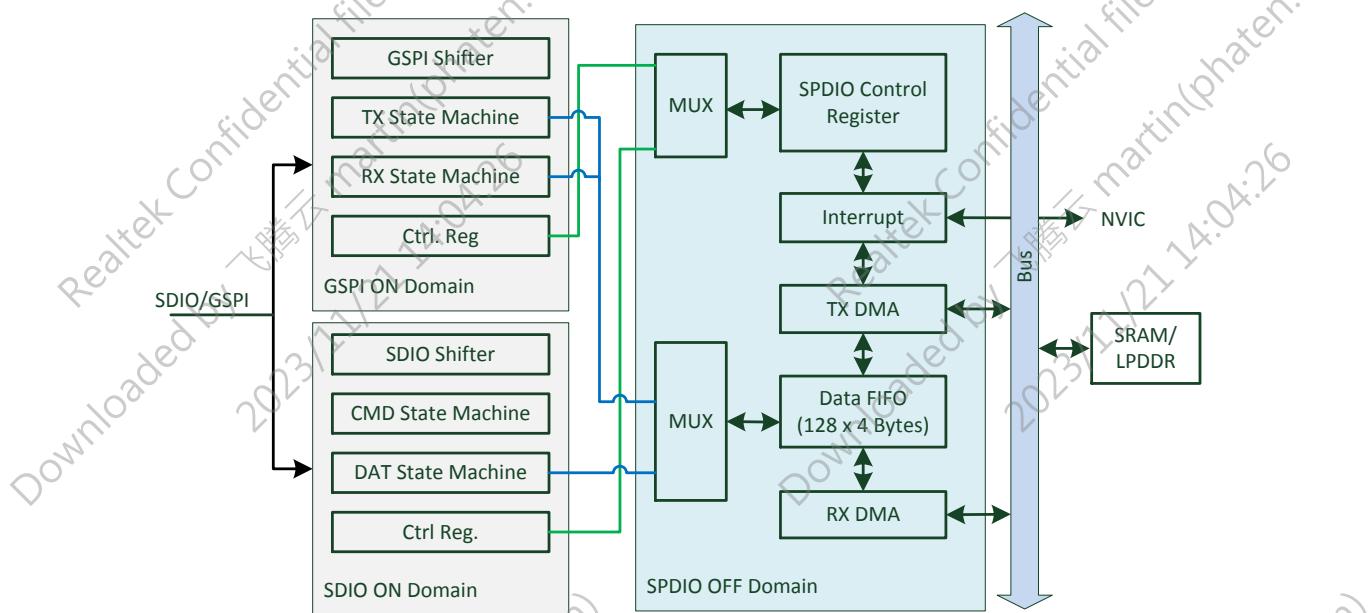


Figure 14-3 SDIO Device/SPI Slave Architecture

14.1.3.1 SDIO Device Mode Architecture

For SDIO Device mode, the equivalent hardware architecture is shown as Figure 14-4. The host side can read/write the control registers via SDIO command CMD52 or CMD53.

The data packets are transferred via SDIO command CMD53. For a SDIO packet transmit, the SDIO host need to read the available receive buffer count register on Ameba-Z II to make sure the Ameba-Z II has memory space to receive this packet. Then the SDIO host sends a CMD53 with this packet to Ameba-Z II. This packet data will be written to the Data FIFO first and then the TX DMA hardware will move these data from the Data FIFO to the given memory buffer. The DMA destination buffer is configured by the Ameba-Z II SDIO/SPI driver.

For a SDIO packet receiving, the SDIO host will start a CMD53 transfer. Ameba-Z II RX DMA will continuously move data from the given memory buffer to the Data FIFO. The Data State Machine will read data from the Data FIFO and shift them to the SDIO bus.

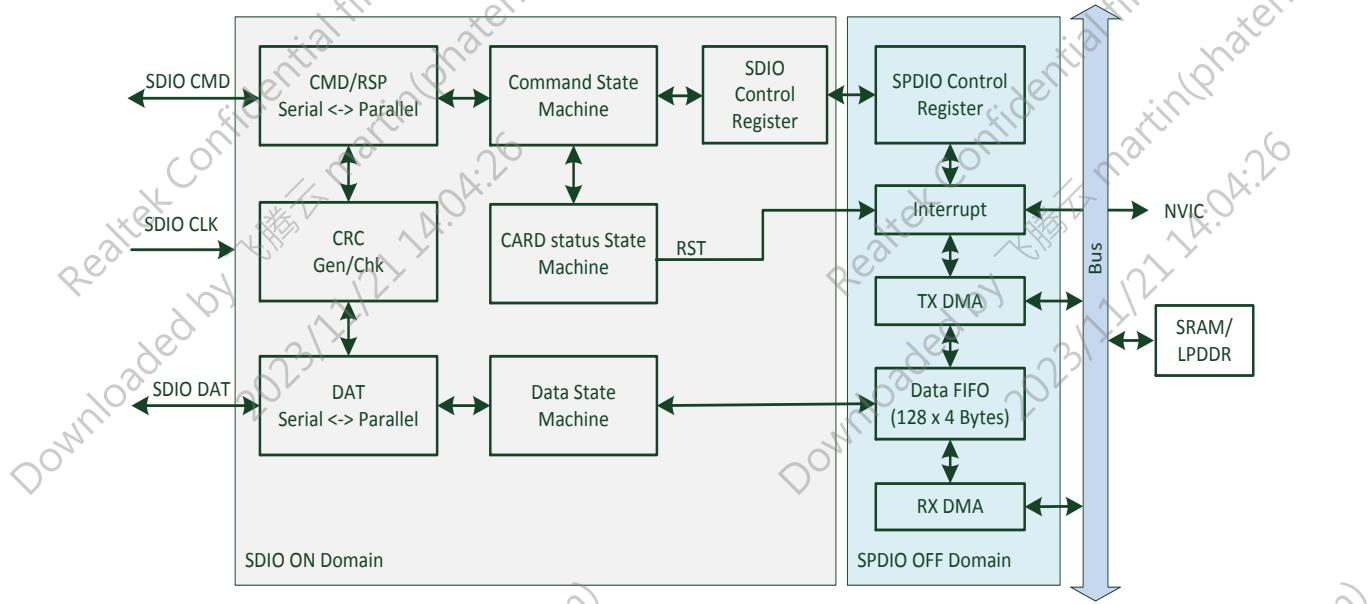


Figure 14-4 SDIO Device Interface Architecture

14.1.3.2 SPI Slave Mode Architecture

For SPI Slave mode, the equivalent hardware architecture is shown as Figure 14-5. The difference with the architecture of SDIO Device hardware is on the bus interface. A Realtek defined proprietary bus protocol is designed for the registers reading/writing and the data packet transmission. The detail of this protocol is described in the following sections.

The data flow of the data packets transmission basically is similar to the SDIO Device interface's behavior.

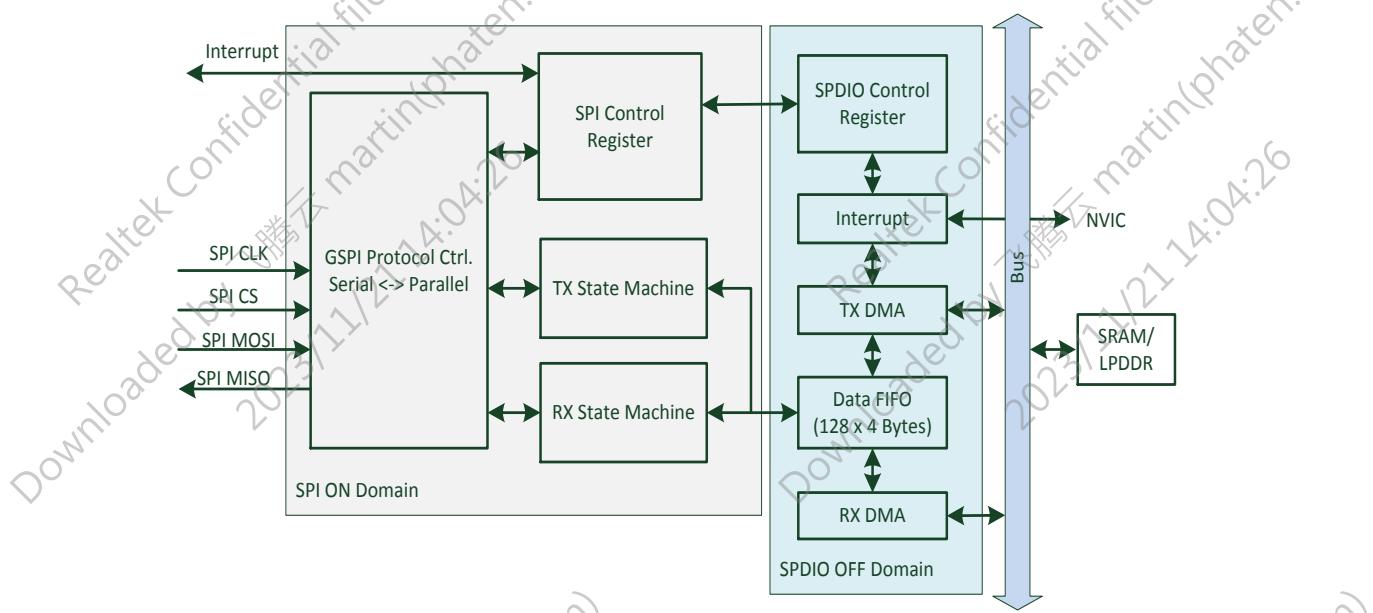


Figure 14-5 SPI Slave Interface Architecture

14.2 Functional Description

The SDIO Device/SPI Slave interface provides a high data rate interface for mass of data transmission. A typical use scenario is Ameba-Z II works as a Wi-Fi network interface device. The application and TCP/IP layer protocol are running on the host system. And then the IP layer packets between the host system and Ameba-Z II are transferred over the SDIO Device/ SPI Slave interface. Figure 14-6 shows an example of software architecture for Ameba-Z II works as a SDIO NIC.

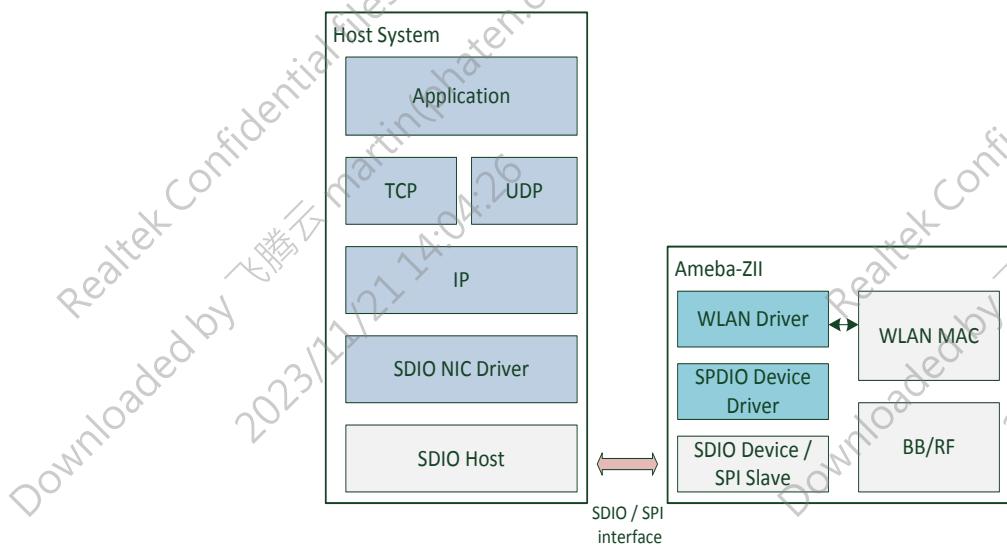


Figure 14-6 Software Architecture Example of Ameba-Z II SDIO NIC

For different application use scenario, user also can define their owned protocol for the data transmission over this SDIO Device/SPI Slave interface. Figure 14-7 shows an example of software architecture for data transmission with user defined protocol over the SDIO Device/SPI Slave interface. In this example the host side application generates data those intend to be transmitted to the internet. The host side application packages those data and follows its defined protocol to transmit them to Ameba-Z II over the SDIO Device/SPI Slave interface. The application software on Ameba-Z II parse packets received from the SDIO Device / SPI Slave interface and then transfers them to the internet over the WLAN.

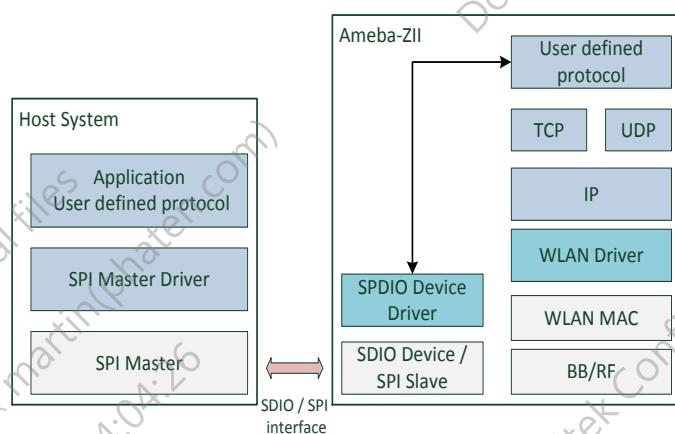


Figure 14-7 Software Architecture Example of User Defined Protocol over SDIO/SPI Interface

14.2.1 SPDIO DMA Transfer

The SDIO Device / SPI Slave interface supports both SDIO bus and SPI bus. However the front end of the data transfer, for both SDIO bus and SPI bus, is processed by hardware purely. The back end of the data transfer is process by the SPDIO OFF domain hardware and Ameba-Z II software. The “SPDIO” is the abbreviation of “SDIO Device / SPI Slave”. From the SPDIO’s point of view, the data transfer flow and behavior is the same, no matter the bus type used for the data transmission.

To reduce the Ameba-Z II CPU loading of the data packets transmission, a DMA hardware is designed to move data from the SPDIO’s data FIFO to a buffer on SRAM or LPDDR of Ameba-Z II platform. This section describes how this DMA works.

14.2.1.1 TX Flow

TX means the data transmission path is from the host side to Ameba-Z II. The data from the host, either through SDIO bus or SPI bus, will be written into the SPDIO’s data FIFO first. Then the TX DMA hardware moves those data from data FIFO to the given buffer.

14.2.1.1.1 TX DMA Configuration

The memory addresses of buffers for storing TX packets data are described by TXBDs (TX Buffer Descriptor). A TXBD actually is 4-bytes memory which is located at SRAM or LPDDR and containing the destination address of a TX DMA transfer for a TX packet data copying. The SPDIO device driver, a software module of Ameba-Z II, is responsible for the TX DMA configuration and the TXBDs initialization.

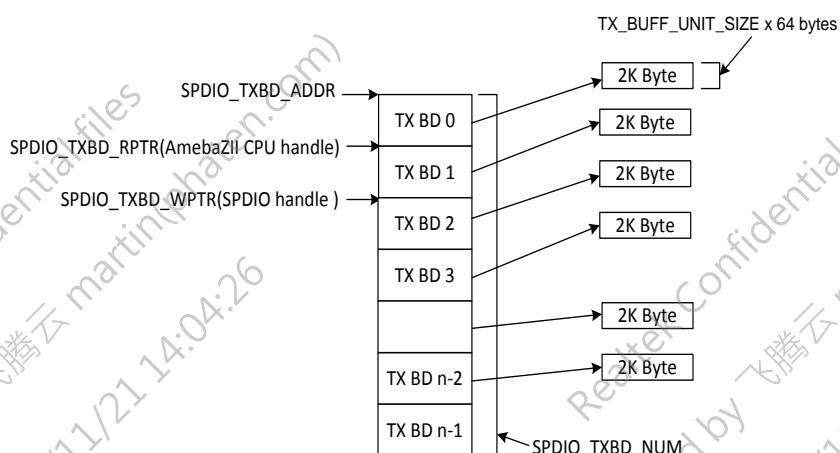


Figure 14-8 SPDIO Tx DMA Configuration

Figure 14-8 is used to illustrate the TX DMA configuration. The SPDIO TX DMA configuration includes:

- Register SPDIO_TXBD_ADDR is used to give the start address of TXBDs. The TX DMA will fetch TX_BDs with an offset from this address. This address must be an 8-bytes aligned address.
- Register SPDIO_TXBD_NUM is used to assign the number of total TXBDs.
- Register TX_BUFF_UNIT_SIZE describes the size of the buffer for TX DMA. The unit is 64 bytes. Ex. TX_BUFF_UNIT_SIZE register value is 24 means the size of the TX DMA buffer is $24 \times 64 = 1536$ bytes.
- Each TXBD point to a TX DMA buffer. The start address of a TX DMA buffer must be an 8-bytes aligned address.
- Register SPDIO_TXBD_WPTR is the index of the TXBD which point to the TX buffer for the next TX packet DMA transfer. This register is maintained by the SPDIO TX DMA hardware state machine. It will be updated as next available TXBD index whenever a TX packet DMA transfers is done. The difference of SPDIO_TXBD_WPTR between SPDIO_TXBD_RPTR is the number of free TX buffer. Once the number of free TX buffer is smaller than 1, the TX DMA hardware will stop working. Until the free TX buffer number is bigger or equal to 1.
- Register SPDIO_TXBD_RPTR is maintained by the SPDIO Device driver (a software module of Ameba-Z II). When processing a TX buffer for a TX DMA transfer is complete, the SPDIO device driver can either modify the corresponding TXBD to make it point to a new TX buffer or reuse the same TX buffer. The SPDIO Device driver need to update this register once the TX packets buffer processing is done. The SPDIO device driver should never let the SPDIO_TXBD_RPTR value run over the SPDIO_TXBD_WPTR value, otherwise some of TX packets will be dropped due to this error.

14.2.1.1.2 TX DMA Behavior

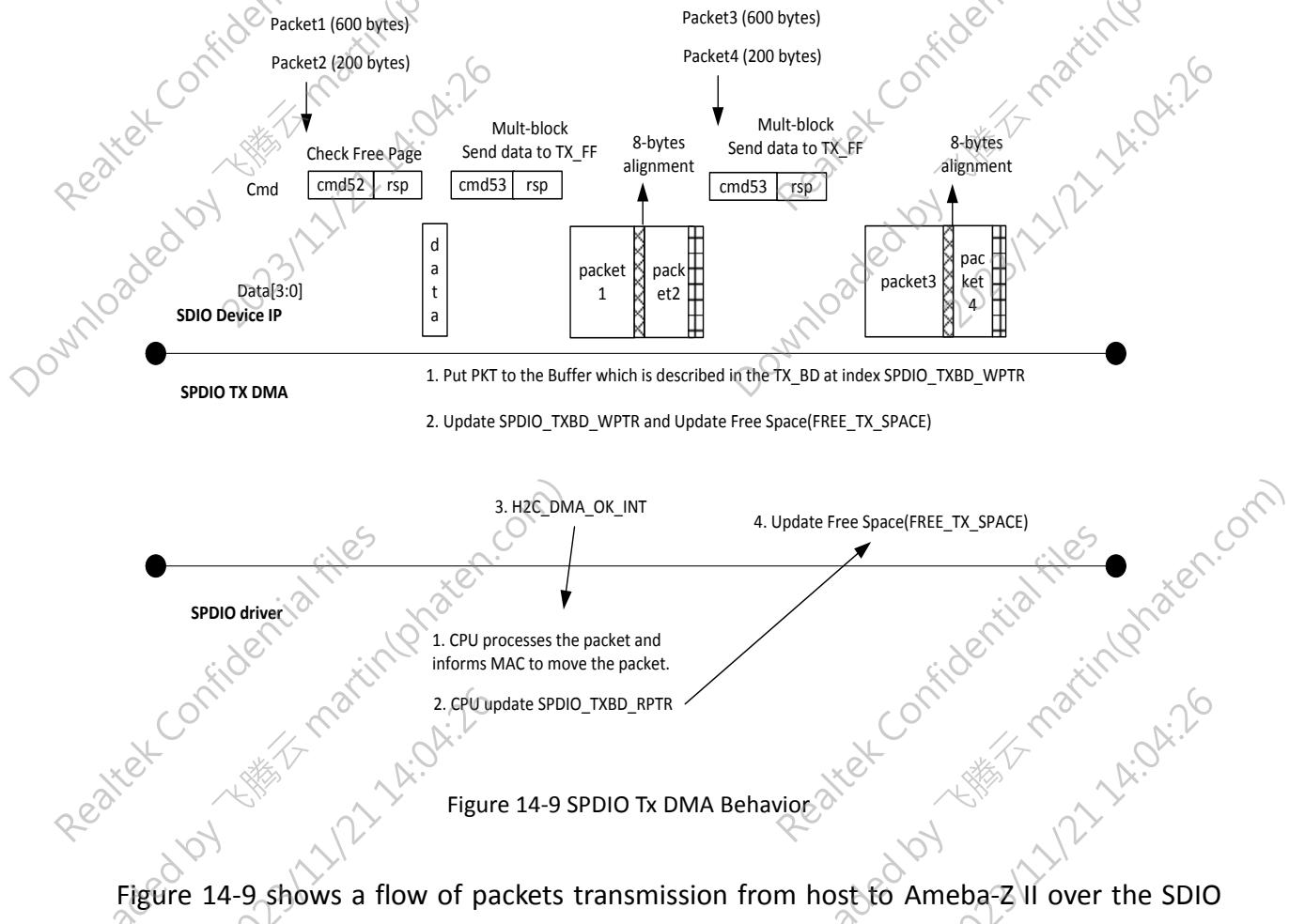


Figure 14-9 SPDIO Tx DMA Behavior

Figure 14-9 shows a flow of packets transmission from host to Ameba-Z II over the SDIO bus to illustrate the SPDIO TX DMA transfer behavior.

On the event of SPDIO receives packet from host:

1. If the available TXBD Number (FREE_TXBD_NUM, which is calculated from SPDIO_TXBD_RPTR and SPDIO_TXBD_WPTR) is bigger than or equal to the packet number received from SDIO host, go to step2, else trigger TXBD_OVF interrupt.
2. If the Packet Size is larger than the TX Buffer Size assigned in register TX_BUFF_UNIT_SIZE, trigger TXPKT_SIZE_OVER_BUFF, else go to next step.
3. Put the packet to the TX Buffer which is described by the TXBD at index SPDIO_TXBD_WPTR.
4. Update SPDIO_TXBD_WPTR and update FREE_TXBD_NUM, which should be updated as long as SPDIO_TXBD_RPTR or SPDIO_TXBD_WPTR changes.
5. Trigger H2C_DMA_OK_INT interrupt to Ameba-Z II CPU to let the SPDIO driver to process received TX packets.

On the event of Ameba-Z II CPU receives H2C_DMA_OK_INT:

1. If SPDIO_TXBD_RPTR equals to SPDIO_TXBD_WPTR (All Packets have been processed by the SPDIO driver), clear H2C_DMA_OK_INT and return; else go to next step.
2. SPDIO driver processes the packet and informs WLAN driver or user's application to move the packet.
3. SPDIO driver updates SPDIO_TXBD_RPTR after the packet processing is finished, go to step1.

14.2.1.2 RX Flow

RX means the data transmission path is from Ameba-Z II to the host side. The RX DMA state machine hardware will copy data packets from the buffer on Ameba-Z II platform to the SPDIO's data FIFO sequentially first, then the SPDIO hardware moves those data from data FIFO to either SDIO Device IP shifter or the SPI Slave IP RX state machine shifter.

14.2.1.2.1 RX DMA Configuration

A RXBD (RX Buffer Descriptor) is used to describe the start memory address of a RX packet data and its size. All of RXBDs should be located in a continuous memory block. The SPDIO device driver is responsible for the RX DMA configuration and the RXBDs initialization. A RX packet can be described by multiple RXBDs. This mechanism is useful to aggregate discontinuous data buffer as a RX packet. The RXBD format is shown as Figure 14-10. And each field is described as the following table:

Field	Offset [bits]	Description
Size	0x00[13:0]	The data size of the RX packet, in byte. This value is valid only when FS bit is 1.
LS	0x00[14]	If the value of this field is 1, it means this RXBD is the last RXBD of a RX packet.
FS	0x00[15]	If the value of this field is 1, it means this RXBD is the first RXBD of a RX packet. The Size field is valid only when this bit value is 1.
Addr	0x04[31:0]	The start address of the RX packet data, it must be an 4-bytes aligned address.

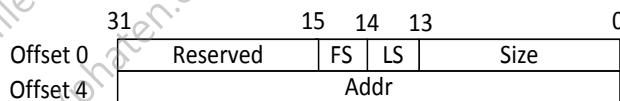


Figure 14-10 SPDIO RxBD Format

Figure 14-11 shows the illustration of the RX DMA configuration. The SPDIO RX DMA configuration includes:

- Register `SPDIO_RXBD_ADDR` is used to give the start address of the memory for all RXBDs. The RX DMA will fetch RX_BDs with an offset from this address. This address must be an 8-bytes aligned address to make the RX DMA can work correctly.
- Register `SPDIO_RXBD_NUM` is used to assign the number of total RXBDs.
- Register `SPDIO_RXBD_WPTR` is maintained by the SPDIO Device driver. When the SPDIO driver sending a RX packet, the SPDIO driver fill one or more of RXBD to describe this RX packet. And then update the `SPDIO_RXBD_WPTR` by increase its value or wrapped as 0 to make its value is the index of the last RXBD of this sending RX packet. The SPDIO device driver should never let the `SPDIO_RXBD_WPTR` value run over the `SPDIO_RXBD_RPTR` value. Otherwise some of RX packets will be dropped due to this error.
- Register `SPDIO_RXBD_RPTR` is the index of the RXBD which describing the RX packet for the RX DMA is going to transfer it to the host side. It is maintained by the SPDIO RX DMA hardware. Once a RX DMA transfer of a RX packet is finished, the RX DMA hardware will update this register to make its value is the index of the last fetched RXBD.

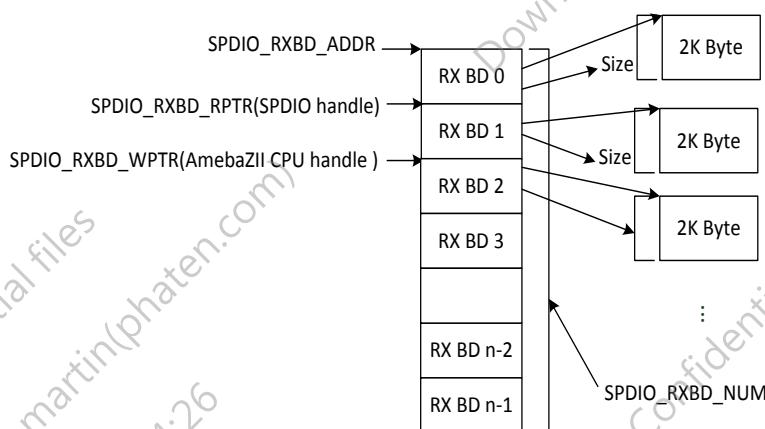


Figure 14-11 SPDIO Rx DMA Configuration

14.2.1.2.2 RX DMA Behavior

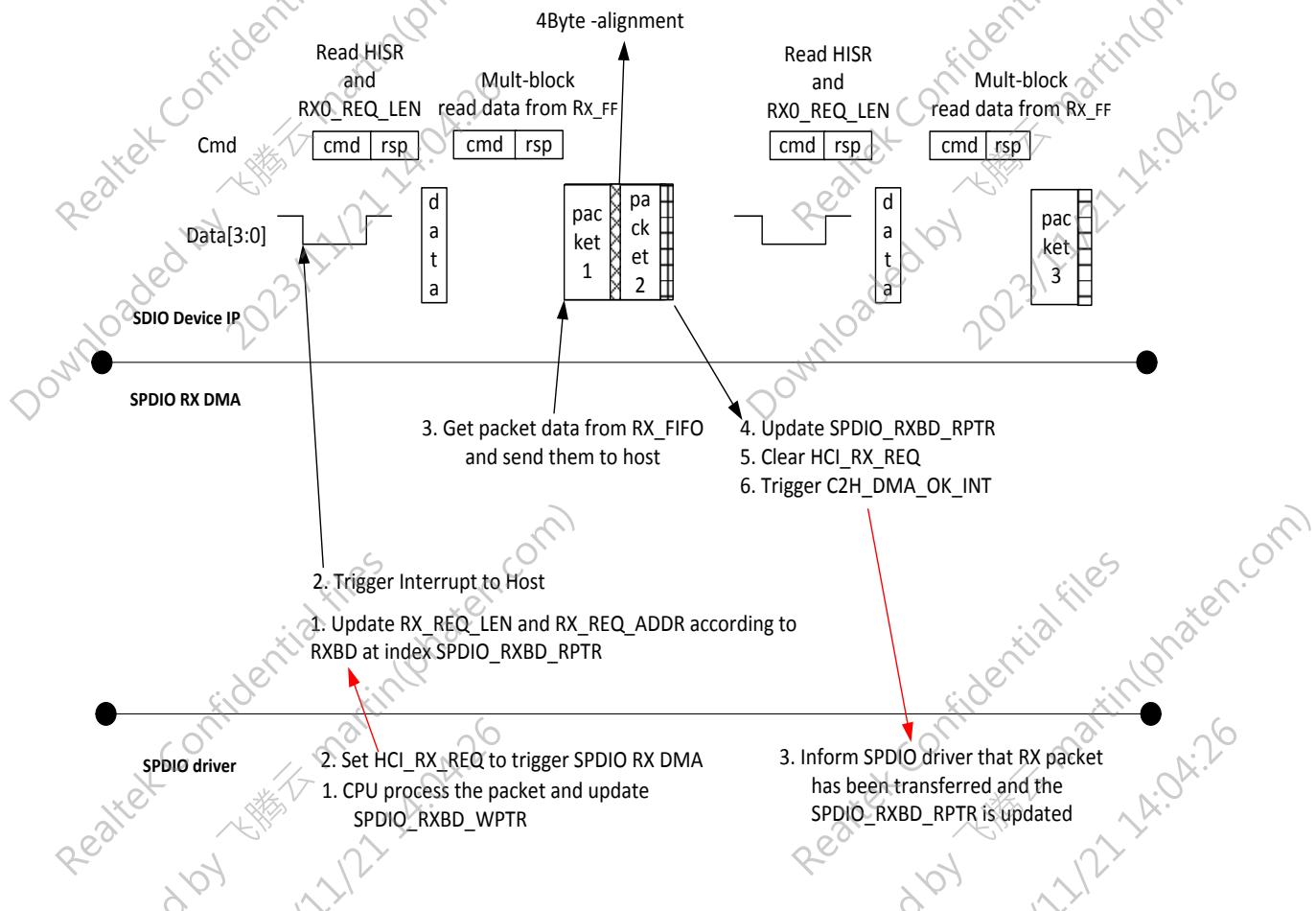


Figure 14-12 SPDIO Rx DMA Behavior

Figure 14-12 shows a flow of packets transmission from Ameba-Z II to host over the SDIO bus to illustrate the SPDIO RX DMA transfer behavior:

On the event of SPDIO Driver is request to send a packet to the host:

1. SPDIO Driver processes this packet and update SPDIO_RXBD_WPTR.
2. Set HCI_RX_REQ to trigger SPDIO RX DMA start the fetching of RXBD.

On the event of SPDIO RX DMA receives HCI_RX_REQ:

1. Update RX_REQ_LEN and RX_REQ_ADDR as described in the RXBD at index SPDIO_RXBD_RPTR.
2. Trigger an interrupt to host to make it start the RX packet transfer.
3. Get packet data from RX_FIFO and sends them to host.
4. SPDIO RX DMA updates the SPDIO_RXBD_RPTR when a packet transfer is finished.
5. If SPDIO_RXBD_RPTR < SPDIO_RXBD_WPTR, go to step1; else go to next step.
6. Clear HCI_RX_REQ.

7. Trigger C2H_DMA_OK_INT interrupt to inform the SPDIO driver that RX packets has been transferred.

On the event of the Ameba-Z II CPU receives C2H_DMA_OK_INT interrupt:

1. Inform the SPDIO driver that RX packets have been transferred and one or more RXBD is free.

14.2.2 SDIO Device Interface Control

The SDIO host side driver can control the SDIO Device hardware via control registers writing. SDIO device IP status and the SPDIO device status also can be got by the registers reading. For packets transmission, the host can use CMD53 with a specific register address to read or write the SDPIO data FIFO.

Here is the “Register Address” coding of SDIO command CMD52 or CMD53 for control register read/write or data FIFO read/write:

Table 14-3 SDIO Register Address

Description	Device ID	Domain ID	Address/Length
Bit	16	[15:13]	[12:0]
Control Register	0	000b	Register address
TX FIFO	0	100b	TX packet length. The SDIO device hardware will check this length to see if the SPDIO device has enough of memory for this packet. If not, this command will be considered as an error command. The packet length includes the TX descriptor length, offset length and the packet payload length. The unit is 4-bytes.
RX FIFO	0	111b	Stuff bits (No used, shall be set to 0)

For packet transmission hardware and software requirement, a packet format is defined as Figure 14-13 shows. This packet format should be applied to all packets on both TX and RX path. To improve the SDIO bus transfer efficient, multiple packets can be aggregated as one packet to be transferred by a single SDIO bus transfer. Figure 14-14 shows an example of packets

aggregation. For TX path, the host side driver is responsible to aggregate multiple packets as one packet. The SPDIO hardware will do the fragment of an aggregation packet. So the SPDIO driver no needs to take care of TX packet aggregation. The SPDIO driver also not needs to do the packets aggregation for the RX path. The SPDIO RX DMA hardware will do the packet aggregation. The host side driver can read SDIO device register RX_REQ_LEN to know the total length of a RX aggregation packet. The host side driver also is responsible to do the fragment of received aggregation packets.

For a TX packet, the start address of the packet should always align to 8-bytes. For a RX packet, the start address of the packet should always align to 4-bytes. When performing packets aggregation, a block of padding data may is needed to be inserted between 2 packets to make all of packets are start with an 8-bytes (for TX) or 4-bytes (for RX) alignment address.

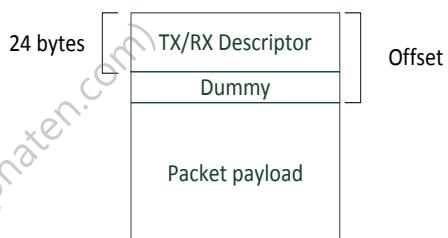


Figure 14-13 Packet Format

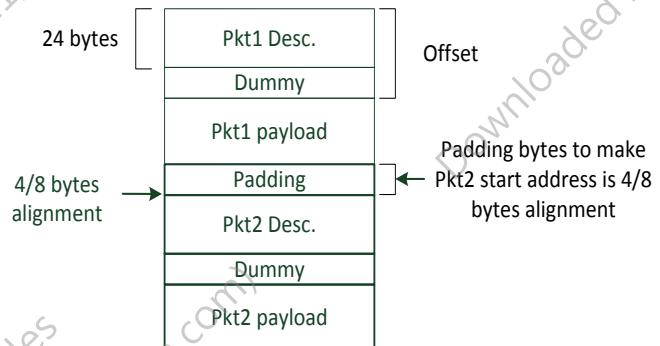


Figure 14-14 Format for Packets Aggregation

All packets should start with a TX Descriptor or RX Descriptor. A TX/RX descriptor is used to describe the packet payload offset, valid packet payload length and the number of packets aggregation.

The size of a TX descriptor is 24-bytes. Following table shows the format of a TX Descriptor and the description of each field:

Table 14-4 TX Descriptor

Bytes Offset	Bits	Field	Description
0	[15:0]	TX Packet Size	The size of a TX packet, in bytes. It includes the size of TX Descriptor, size of dummy bytes, and the size of a packet payload. For the 1st TX descriptor of an aggregation packet, this size includes all the data in this packet (all TX descriptors, dummy, payload of all packets and all padding bytes).
0	[23:16]	Offset	The offset from the packet start to the payload start. If no dummy data exist, the offset should be 24 bytes (the size of a TX descriptor).
0	[31:24]	Agg. Number	The number of packets of a packet aggregation. For an aggregation packet, this field is valid for the 1 st TX descriptor only. For a non-aggregation packet, this field should be set to 1.
4	[7:0]	Packet Type	The type of a packet. It is defined by SDIO device driver implementation.
4	[31:8]	Reserved0	Reserved for software.
8	[31:0]	Reserved1	Reserved for software.
12	[31:0]	Reserved2	Reserved for software.
16	[31:0]	Reserved3	Reserved for software.
20	[31:0]	Reserved4	Reserved for software.

The size of a RX descriptor also is 24-bytes. Following table shows the format of a RX Descriptor and the description of each field:

Table 14-5 RX Descriptor

Offset	Bytes	Bits	Field	Description
0	[15:0]		RX Packet Size	The size of a RX packet, in bytes. It includes the size of RX Descriptor, size of dummy bytes, and the size of a packet payload.
0	[23:16]		Offset	The offset from the packet start to the payload start. If no dummy data exist, the offset should be 24 bytes (the size of a RX descriptor).
0	[31:24]		Reserved0	Reserved for software.
4	[7:0]		Packet Type	The type of a packet. It is defined by SDIO device driver implementation.
4	[31:8]		Reserved1	Reserved for software.
8	[31:0]		Reserved2	Reserved for software.
12	[31:0]		Reserved3	Reserved for software.
16	[31:0]		Reserved4	Reserved for software.
20	[31:0]		Reserved5	Reserved for software.

14.2.3 SPI Slave Interface

The SPI Slave IP provides a high data rate interface for Ameba-Z II to communicate with other system (host system). Figure 14-15 shows how the SPI bus connection between Ameba-Z II and a host system.

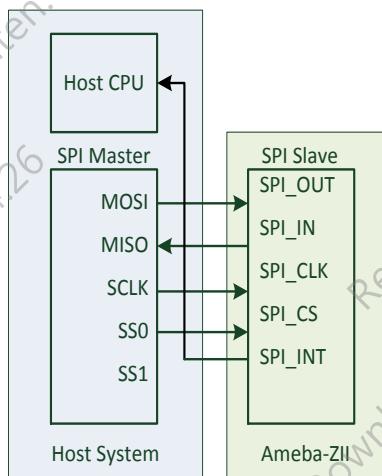


Figure 14-15 The Connection of Ameba-Z II SPI Slave with Host System

This SPI Slave interface is fully compliant with general SPI bus behavior. The SPI Slave interface pins and their description are listed as the following table:

Table 14-6 SPI Slave Pins

Pin name	Description
SPI_IN	Input to SPI master and output from Ameba-Z II
SPI_OUT	Output from SPI master and input to Ameba-Z II
SPI_CLK	SPI clock (output from SPI master)
SPI_CS	SPI chip select, active low (output from SPI master)
SPI_INT	The extra pin to send interrupt signal to the host, active low output(output from Ameba-Z II)

The SPI bus timing waveform is shown as Figure 14-16. A high-to-low transition on the SPI_CS pin is required to start an operation, and a low-to-high transition is required to end an operation. SPI_IN and SPI_OUT are falling driving and rising latch. Commands and data are shifted out with sequence of MSB first and LSB last.

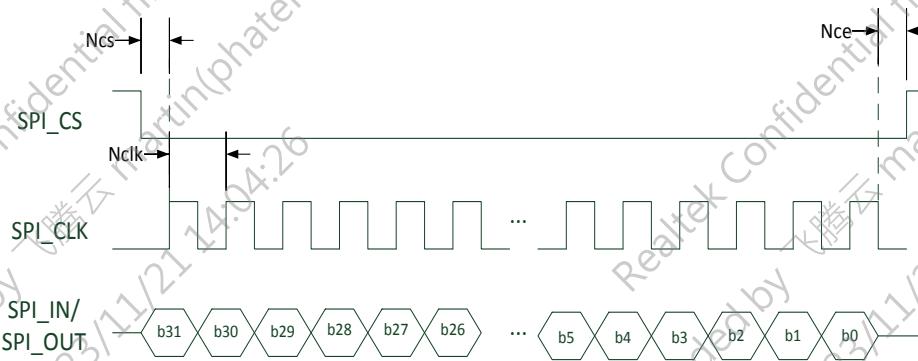


Figure 14-16 The Timing Waveform for Ameba-Z II SPI Slave Interface

Timing	Min.	Max.
Nclk	-	50 MHz
Ncs	20 ns	-
Nce	20 ns	-

14.2.3.1 SPI Interface Protocol

Realtek defines a proprietary protocol on this SPI Slave interface. This SPI protocol supports both 16-bit and 32-bit word operation. Also supports both Little-Endian and Big-Endian operation. A SPI transfer is divided as 3 stages: Command, Status and Data. The description of each stage is list as the table below:

Table 14-7 SPI Transfer Command Stage

Stage	Description
Command	A SPI transfer always is started with a 4-bytes Command from the SPI master. The command can be a SPI control register reading/writing or SPDIO data FIFO reading/writing.
Status	The Ameba-Z II will send an 8-bytes status in this Status stage. The SPI Slave hardware will collect some of internal status from its control registers. This can reduce the SPI control register polling from the software of host side.
Data	The data transfer of Data stage is use to convey register value or the packet data. The data unit is 32-bits size. For a register/RX_FIFO reading command, the data is sent from Ameba-Z II to the SPI master. For a register/TX_FIFO writing command, the data is sent from the SPI master to Ameba-Z II.

The protocol for SPI control register reading is Status follows Command. Status is followed by Data. Figure 14-17 shows the protocol for register reading.

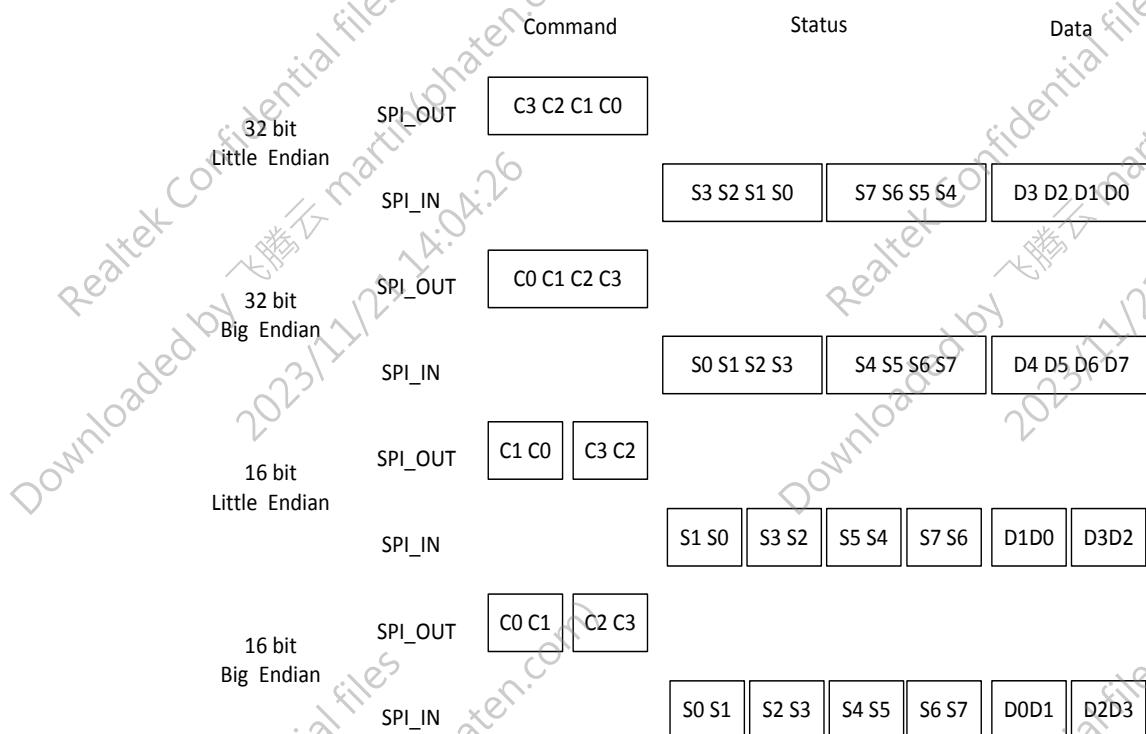


Figure 14-17 The SPI Slave Interface Protocol for Register Reading

The protocol for SPI control register writing is Data follows Command. And Data is followed by Status. It's shown as Figure 14-18.

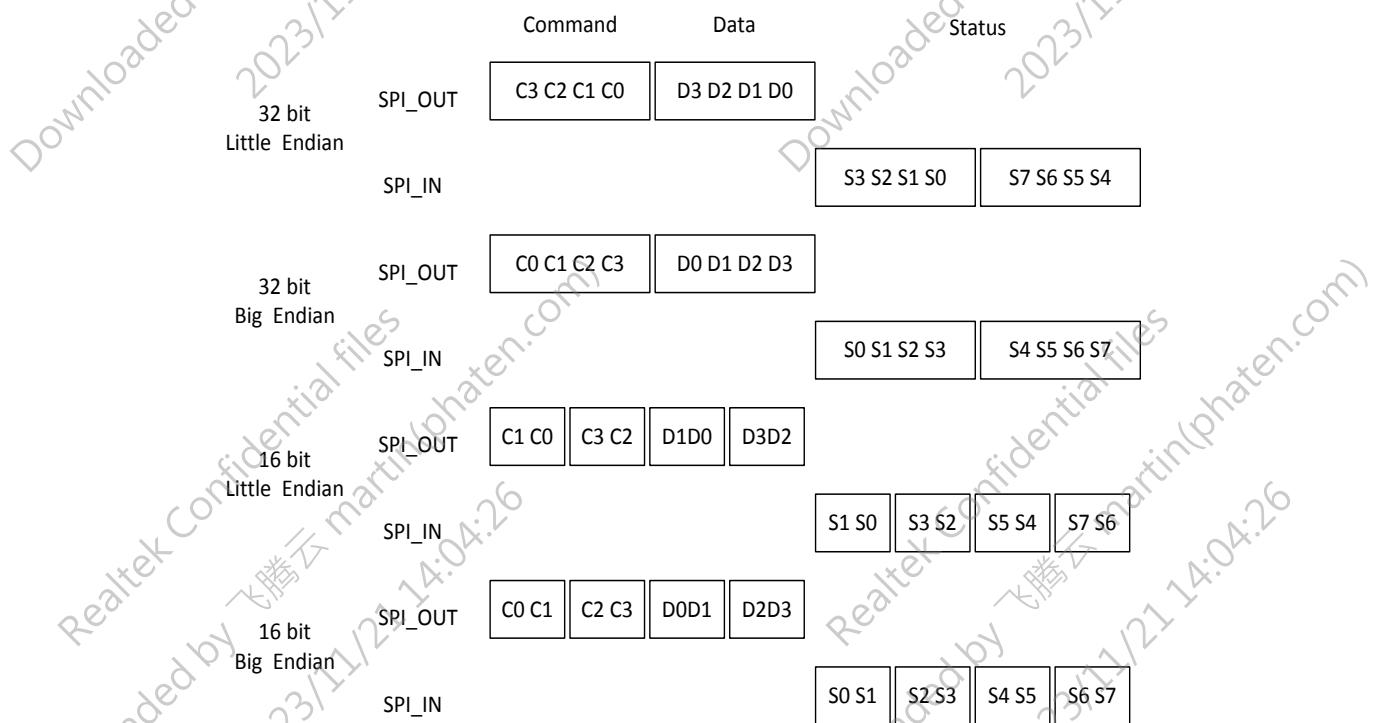


Figure 14-18 The SPI Slave Interface Protocol for Register Writing

The protocol for packet RX (SPDIO RX_FIFO reading) is similar to the protocol for register reading. It's shown as Figure 14-19.

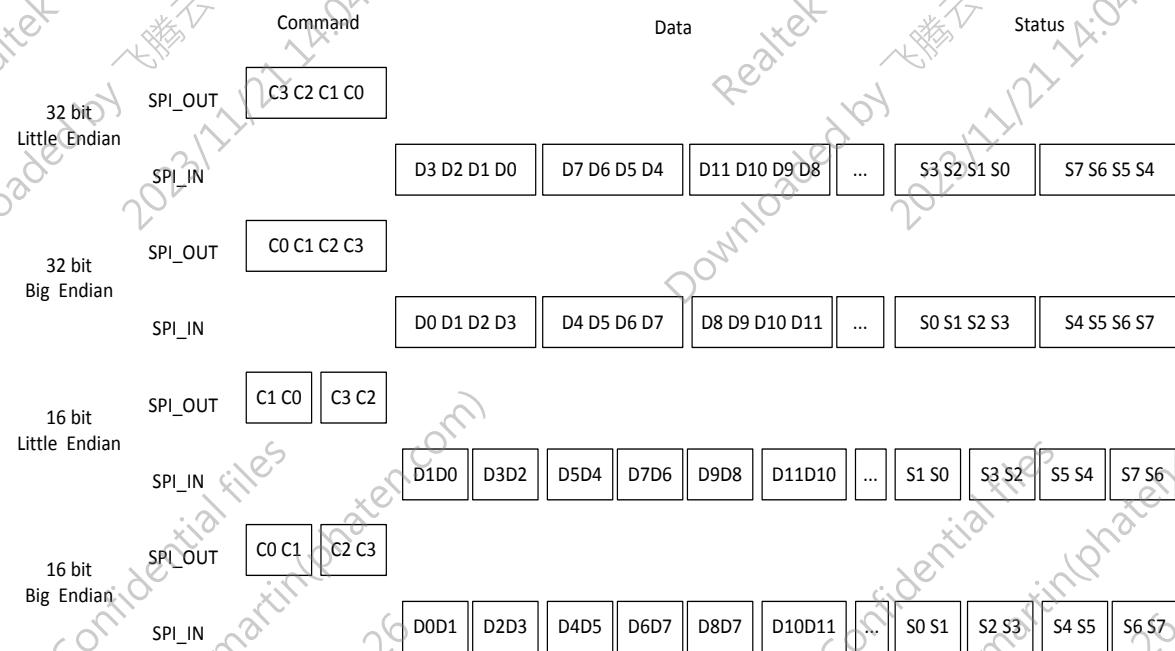


Figure 14-19 The SPI Slave Interface Protocol for Packet Rx (RX_FIFO reading)

The protocol for packet TX (SPDIO TX_FIFO writing) is similar to the protocol for register writing. It's shown as Figure 14-20.

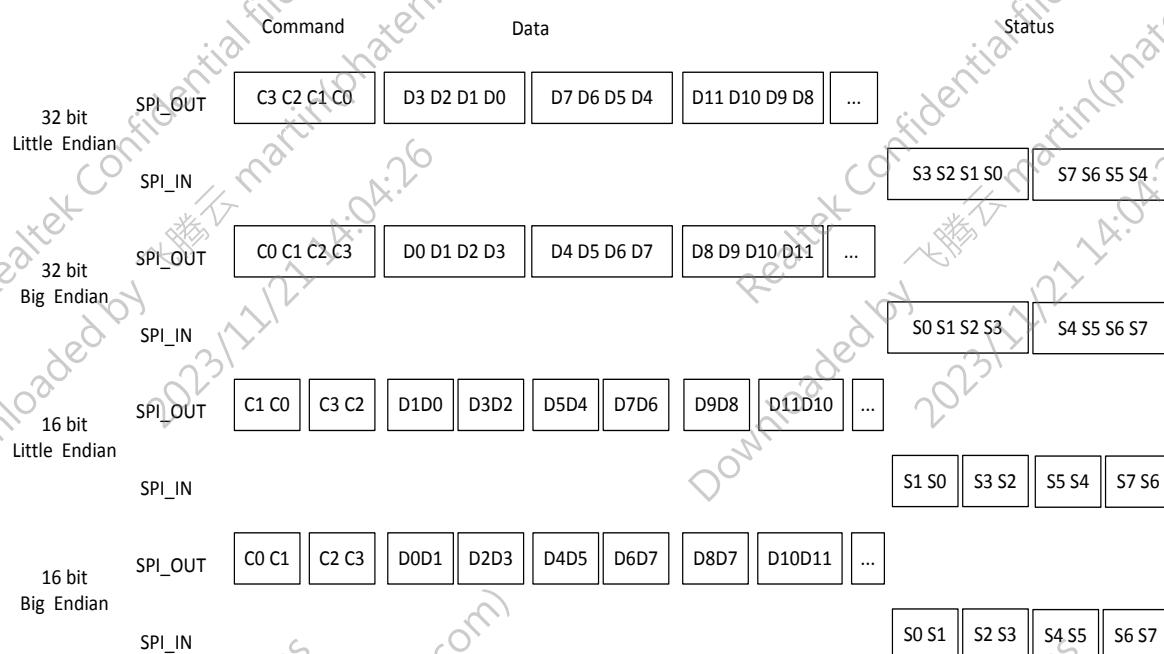


Figure 14-20 The SPI Slave Interface Protocol for Packet Tx (TX_FIFO writing)

14.2.3.1.1 SPI Command Format

The format of SPI Command is shown as Figure 14-21.

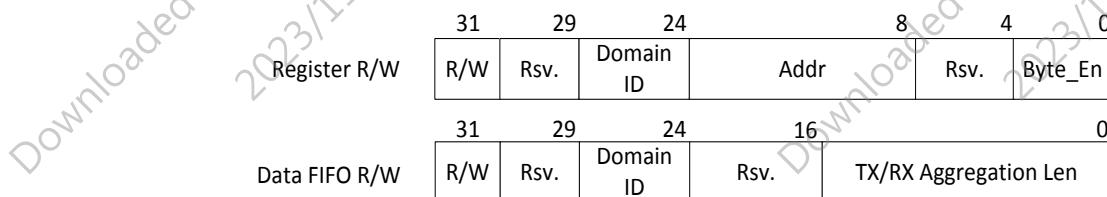


Figure 14-21 The SPI Command Format

Table 14-8 Format of Control Register and Data FIFO

Description	Domain id		Addr.		length
Bit	28:26	25:24	23:8		7:0
Ctrl. Register	000	xx	0x0000 ~ 0xFFFF		RSV
Data FIFO	011	TX_FIFO (00b)	RSV[23:16]	TX Aggregation Len[15:0]	
	111	RX_FIFO (11b)	RSV[23:16]	RX Aggregation Len[15:0]	

The description of each command field is listed in the following table:

Table 14-9 Command Fields

Field	Bit	Description
R/W	31	This field Indicates the data direction. 0: it's a command for register or RX_FIFO reading; 1: it's a command for register or TX_FIFO writing.
Domain ID	[28:24]	This field indicated the domain for this command's operation. 00000b: It's a command for SPI control register reading or writing. 01100b: It's a command for SPDIO TX_FIFO writing. 11111b: It's a command for SPDIO RX_FIFO reading.
Addr.	[23:8]	The address offset of the control register to be written/read by this command. The address offset is 4-bytes alignment.
Byte_En	[3:0]	For a register writing command, Byte_En tells which bytes will be written. 0001b: the first byte (LSB) will be written. 0010b: the second byte will be written. 0100b: the third byte will be written 1000b: the forth byte (MSB) will be written 0011b: the first byte and the second byte will be written 1100b: the third and the forth byte will be written. 1111b: all four bytes will be written.
TX/RX Aggregation Len	[15:0]	The field contains the number of bytes to write TX_FIFO or read RX_FIFO.

14.2.3.1.2 SPI Status Format

The SPI Status is used to returns the SPI Slave's current status. This status provides the host side driver needed information. For example, the host side driver can get free TXBD number from the status of previous TX packet transfer. So the host side can accord to this free TXBD number to arrange further packet transmission. The SPI Slave hardware collects information from SPI control register to form this status. The format of the status is shown as Figure 14-22.

Reg. 0x1C		Reg. 0x20		Reg. 0x18	
Name	State	RX_REQ_LEN	FREE_TX_BD_NUM	HISR1	HISR0
Bit size	8	24	27	3	2

Figure 14-22 The SPI Status Format

The description of each SPI Status field is listed in the following table.

Field	Bit	Description
State	[63:56]	The value of this field is get from SPI Register RX0_REQ_LEN[31:24]. Please reference the SPI register for the definition of this field.
RX_REQ_LEN	[55:32]	The value of this field is get from SPI Register RX0_REQ_LEN[23:0]. The field indicates the pending RX Packet length on SPDIO, unit is byte. For the RX aggregation case, the length is total length of the packet.
FREE_TX_BD_NUM	[31:5]	The value of this field is get from SPI Register FREE_TX_BD_NUM[28:2]. This field indicates the free TXBD number of

		SPDIO TX DMA.
HISR1	[4:2]	The value of this field is get from SPI Register SPI_HISR[19:17]. Please reference the SPI register for the definition of this field.
HISRO	[1:0]	The value of this field is get from SPI Register SPI_HISR[1:0]. Please reference the SPI register for the definition of this field.

14.3 SDIO Control Registers

14.3.1 SDIO_TX_CTRL

- Name: SDIO_TX_CTRL Register
- Width: 8bit
- Initial Value: 0x14

REG 14-10 (Offset 0000h) SDIO_TX_CTRL

Bit	Access	INI	Symbol	Description
7:6				RSVD
5	R/W		SDIO_CMD_FORCE_VLD	Force ALL valid command are legal at any card state.
4	R/W		R_INIT_CMD_EN	0:disable; 1:enable init command response(CMD 0 5 3 7) even after power on init Autoload from Efuse 0xD3[4]
3	R/W		R_EN_32K_TRANS	
2	R/W		EN_RXDMA_MASK_INT	If this bit set to 0, RX_REQUEST interrupt will raised immediately when RX data is ready. If this bit set to 1 and status information is sent, RX_REQUEST interrupt will not raised.
1	R/W		EN_MASK_TIMER	The bit is only valid when EN_RXDMA_MASK_INT is 1. If this bit set to 0, timeout is disable. If this bit set to 1, it enable interrupt timeout.
0	R/W		CMD_ERR_STOP_EN	

14.3.2 SDIO_STATIS_RECOVERY_TIMEOUT

- Name: SDIO_STATIS_RECOVERY_TIMEOUT Register
- Width: 16bit
- Initial Value: 0x0C00

REG 14-11 (Offset 0002h) SDIO_STATIS_RECOVERY_TIMEOUT

Bit	Access	INI	Symbol	Description
15:0	R/W		SDIO_STATIS_RECOVERY_TIME_OUT	It is SDIO status timeout. The unit is 32usec. The default value is 0xc00 means that timeout is 98304 usec (32*0xc00)

14.3.3 SDIO_HIMR

- Name: SDIO_HIMR Register
- Width: 32bit
- Initial Value: 0xF

REG 14-12 (Offset 0014h) SDIO_HIMR

Bit	Access	INI	Symbol	Description
31:23	R/W		NO USED	NO USED
22	R/W		CPU_NOT_RDY_MSK	This bit is clear, and CPU_NOT_RDY is masked.
21	R/W		NO USED	NO USED
20	R/W		H2C_BUS_FAIL_MSK	This bit is clear, and H2C_BUS_FAIL_INT is masked
19	R/W		CPWM2_MSK	This bit is clear, and CPWM2_INT is masked
18	R/W		CPWM1_MSK	This bit is clear, and CPWM1_INT is masked
17	R/W		C2H_MSG_MSK	This bit is clear, and C2H_MSG_INT is masked
16:5	R/W		NO USED	NO USED
4	R/W		TXBD_OVERFLOW_MSK	This bit is clear, and TXBD_OVERFLOW is masked
3	R/W		TXAGG_SIZE_MISMATCH_MSK	This bit is clear, and TXAGG_SIZE_MISMATCH is masked
2	R/W		TXPKT_SIZE_OVER_BUFF_MSK	This bit is clear, and TXPKT_SIZE_OVER_BUFF is masked
1	R/W		TXFIFO_AVAL_MSK	This bit is clear, and TXFIFO_AVAL_INT INT is masked
0	R/W		RX_REQUEST_MSK	This bit is clear, and RX_REQUEST_INT is masked.

14.3.4 SDIO_HISR

- Name: SDIO_HISR Register
- Width: 32bit
- Initial Value: 0x0

REG 14-13 (Offset 0018h) SDIO_HISR

Bit	Access	INI	Symbol	Description
31:21			NO USED	NO USED
22	R/W1C		CPU_NOT_RDY	When host send CMD53 for to access SDIO OFF, and CPU_RDY_IND=0, trigger this interrupt
21			NO USED	NO USED
20	R/W1C		H2C_BUS_FAIL	H2C BUS Fail Interrupt: To notify Host DMA transfer error by AHB BUS resource conflict. Host get this error INT, and know the previous packet fail, and need to re-send or IO_RESET device!
19	R/W1C		CPWM2_INT	CPWM2 Write Interrupt: This interrupt will be raised when FW writes CPWM2_TOGGLING of CPWM2 register. Write 1 clear
18	R/W1C		CPWM1_INT	CPWM1 Write Interrupt: This interrupt will be raised when FW writes CPWM1_TOGGLING of CPWM1 register. Write 1 clear
17	R/W1C		C2H_MSG_INIT	CPU to Host Message INT Status: This interrupt will be raised when FW sends a C2H MSG

			Write 1 clear
16:5		NO USED	NO USED
4	R/W1C	TXBD_OVF	When host send TX packet and there is no enough TXBD, trigger this interrupt
3	R/W1C	TX_AGG_SIZE_MISMATCH	If the total size of the tx aggregation packets is LARGER the CMD53 Data Size, this interrupt trigger
2	R/W1C	TXPKT_SIZE_OVER_BUFF	If the Packet size (Aggregated or not) is larger than the Buffersize assigned in the TX_BD, this interrupt trigger
1	R/W1C	TXAVAL_INT	When the free page is lower than SDIO_AVAIL_NUM_TH_L and higher than SDIO_AVAIL_NUM_TH_H consequently, this interrupt occurs. When Host read FREE_TXBD_NUM, the bit becomes zero. When using CMD52 to read FREE_TXBD_NUM, we suggest to read offset 0x23->0x22->0x21->0x20 consequently
0	R	RX_REQUEST	This bit is set when one complete RX aggregation packet is received into RXPKTBUF and RX_REQUEST_MSK is set to 1. Only when RX FIFO is empty, the bit becomes zero.

14.3.5 RX0_REQ_LEN

- Name: RX0_REQ_LEN Register
- Width: 32bit
- Initial Value: 0x0

REG 14-14 (Offset 001Ch) RX0_REQ_LEN

Bit	Access	INH	Symbol	Description
31	R		RX_REQ_LEN_RDY	Indicate RX0_REQ_LEN is ready
30	R		TXDMA_FIFO_OK	This bit is usage for GSPI mode, for GSPI need polling this bit to confirm previous TX transaction is from FIFO to SRAM and then start next NEW transaction. (could assign TXFIFO_WLEVEL to 1 for avoid polling bit)
29:24			RSVD	
23:0	R		RX0_REQ_LEN	The requested RX Packet length; Unit: Byte For the RX Aggregation case, the length is total length of the packet. Each packet is 8byte 4byte alignment. This value is legal only when RX_REQ_LEN_RDY is 1

14.3.6 FREE_TXBD_NUM

- Name: FREE_TXBD_NUM Register
- Width: 16bit
- Initial Value: 0x0

REG 14-15 (Offset 0020h) FREE_TXBD_NUM

Bit	Access	INI	Symbol	Description
15:0	R		FREE_TXBD_NUM	The free TXBD Number.

14.3.7 TX_SEQNUM

- Name: TX_SEQNUM Register
- Width: 8bit
- Initial Value: 0x0

REG 14-16 (Offset 0024h) TX_SEQNUM

Bit	Access	INI	Symbol	Description
7:0	R		TX_SEQNUM	NO USED

14.3.8 HCPWM

- Name: HCPWM Register
- Width: 8bit
- Initial Value: 0x0

REG 14-17 (Offset 0038h) HCPWM

Bit	Access	INI	Symbol	Description
7	R		TOGLGING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
6:4	R		NO USED	NO USED
3	R		WWLAN	1: Wake On WLAN State; 0: Normal State;
2	R		RPS_ST	1: AP Register Active State; 0: AP Register Sleep State;
1	R		WLAN_TRX	1: WLAN On; 0: WLAN Off.
0	R		RSVD	

14.3.9 HCPWM2

- Name: HCPWM2 Register
- Width: 16bit
- Initial Value: 0x0

REG 14-18 (Offset 003Ah) HCPWM2

Bit	Access	INI	Symbol	Description
15	R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
14:0	R		NO USED	NO USED

14.3.10 REG_SDIO_H2C_MSG

- Name: REG_SDIO_H2C_MSG Register
- Width: 32bit
- Initial Value: 0x0

REG 14-19 (Offset 0060h) REG_SDIO_H2C_MSG

Bit	Access	INI	Symbol	Description
31	R/W		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
30:0	R/W		REG_SDIO_H2C_MSG	Driver to FW Message

14.3.11 REG_SDIO_C2H_MSG

- Name: REG_SDIO_C2H_MSG Register
- Width: 32bit
- Initial Value: 0x0

REG 14-20 (Offset 0064h) REG_SDIO_C2H_MSG

Bit	Access	INI	Symbol	Description
31	R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
30:0	R		REG_SDIO_C2H_MSG	FW to Driver Message

14.3.12 REG_SDIO_H2C_MSG_EXT

- Name: REG_SDIO_H2C_MSG_EXT Register
- Width: 32bit
- Initial Value: 0x0

REG 14-21 (Offset 0068h) REG_SDIO_H2C_MSG_EXT

Bit	Access	INI	Symbol	Description
31:0	R/W		REG_SDIO_H2C_MSG_EXT	Driver to FW EXT Message, trigger by 0x0060[31]

14.3.13 REG_SDIO_C2H_MSG_EXT

- Name: REG_SDIO_C2H_MSG_EXT Register
- Width: 32bit
- Initial Value: 0x0

REG 14-22 (Offset 006Ch) REG_SDIO_C2H_MSG_EXT

Bit	Access	INI	Symbol	Description
31:0	R		REG_SDIO_C2H_MSG_EXT	FW to Driver EXT Message

14.3.14 HRPWM

- Name: HRPWM Register
- Width: 8bit
- Initial Value: 0x7

REG 14-23 (Offset 0080h) HRPWM

Bit	Access	INI	Symbol	Description
7	R/W		TOGLLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
6:4	R/W		NO USED	
3	R/W		WWLAN	1: Wake On WLAN State; 0: Normal State;
2	R/W		RPS_ST	1: AP Register Active State; 0: AP Register Sleep State;
1	R/W		WLAN_TRX	1: WLAN On; 0: WLAN Off.
0	R/W		RSVD	

14.3.15 HRPWM2

- Name: HRPWM2 Register
- Width: 16bit
- Initial Value: 0x0

REG 14-24 (Offset 0082h) HRPWM2

Bit	Access	INI	Symbol	Description
15	R/W		TOGLGING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
14:0	R/W		NO USED	NO USED

14.3.16 HPS_CLKR

- Name: HPS_CLKR Register
- Width: 8bit
- Initial Value: 0x0

REG 14-25 (Offset 0084h) HPS_CLKR

Bit	Access	INI	Symbol	Description
7:0	R/W		NO USED	NO USED

14.3.17 CPU_INDICATION

- Name: CPU_INDICATION Register
- Width: 8bit
- Initial Value:

REG 14-26 (Offset 0087h) CPU_INDICATION

Bit	Access	INI	Symbol	Description
7:4	R/W		SDIO_TEST_BUSY_CNT	For Device test BUSY control count, BUSY time \approx BUSY_CNT * 16-SD_CLK
3:1				RSVD
0	R		CPU_RDY_IND	Used to indicate that CPU is ready for TRX. Default: 0 This bit is synchronized from SYSTEM_CPU_RDY_IND(Offset 0x1C5)

14.3.18 CMD_IN_TO_RSP_OUT_TIME

- Name: Command IN to Response OUT Time Register
- Width: 16bit
- Initial Value: 0xA

REG 14-27 (Offset 0088h) CMD_IN_TO_RSP_OUT_TIME

Bit	Access	INI	Symbol	Description
15:0	R/W		CMDIN_2RESP_TIME	

14.3.19 ERR_FLAG

- Name: ERR_FLAG Register
- Width: 8bit
- Initial Value: 0x0

REG 14-28 (Offset 00C0h) ERR_FLAG

Bit	Access	INI	Symbol	Description
7:0	R/W1C		ERR_FLAG	ERR FLAG for SDIO DEBUG

14.3.20 SDIO_CMD_ERRCNT

- Name: SDIO_CMD_ERRCNT Register
- Width: 8bit
- Initial Value: 0x0

REG 14-29 (Offset 00C2h) SDIO_CMD_ERRCNT

Bit	Access	INI	Symbol	Description
7:0	R/W		CMD_CRC_ERR_CNT	The number of CRC error of SDIO Command

14.3.21 SDIO_DATA_ERRCNT

- Name: SDIO_DATA_ERRCNT Register
- Width: 8bit
- Initial Value: 0x0

REG 14-30 (Offset 00C3h) SDIO_DATA_ERRCNT

Bit	Access	INI	Symbol	Description
7:0	R/W		DATA_CRC_ERR_CNT	The number of CRC error of SDIO Data Block

14.3.22 SDIO_CRC_ERR_INDEX

- Name: SDIO_CRC_ERR_INDEX Register
- Width: 8bit
- Initial Value: 0x0

REG 14-31 (Offset 00C4h) SDIO_CRC_ERR_INDEX

Bit	Access	INI	Symbol	Description
7:5				RSVD
4	R/W1C		D3_CRC_ERR	Indicate there is crc error on SD_D3 It is suggested to clear it by CMD52
3	R/W1C		D2_CRC_ERR	Indicate there is crc error on SD_D2 It is suggested to clear it by CMD52
2	R/W1C		D1_CRC_ERR	Indicate there is crc error on SD_D1 It is suggested to clear it by CMD52
1	R/W1C		D0_CRC_ERR	Indicate there is crc error on SD_D0 It is suggested to clear it by CMD52
0				RSVD

14.3.23 SDIO_AVAI_BD_NUM_TH_L

- Name: SDIO_AVAI_BD_NUM_TH_L Register
- Width: 32bit
- Initial Value: 0x0

REG 14-32 (Offset 00D0h) SDIO_AVAI_BD_NUM_TH_L

Bit	Access	INI	Symbol	Description
31:16				RSVD
15:0	R/W		SDIO_AVAI_BD_NUM_TH_L	Low threshold for TXBD Number

14.3.24 SDIO_AVAI_BD_NUM_TH_H

- Name: SDIO_AVAI_BD_NUM_TH_H Register
- Width: 32bit
- Initial Value: 0x0

REG 14-33 (Offset 00D4h) SDIO_AVAI_BD_NUM_TH_H

Bit	Access	INI	Symbol	Description
31:16				RSVD
15:0	R/W		SDIO_AVAI_BD_NUM_TH_H	High threshold for TXBD Number

14.3.25 SDIO_RX_AGG_CFG

- Name: SDIO_RX_AGG_CFG Register
- Width: 16bit
- Initial Value: 0x0

REG 14-34 (Offset 00D8h) SDIO_RX_AGG_CFG

Bit	Access	INI	Symbol	Description
15	R/W		SDIO_RX_AGG_EN	Enable SDIO Bus RX Aggregation
14:8	R/W		RX_AGG_TO	Timeout threshold for RX Agg; Unit: 32us
7:0	R/W		RX_AGG_BD_COUNT_TH	RX_BD count threshold for RX Agg

14.3.26 SDIO_CMD_TIMING_CTRL

- Name: SDIO_CMD_TIMING_CTRL Register
- Width: 8bit
- Initial Value: 0x0

REG 14-35 (Offset 00DCh) SDIO_CMD_TIMING_CTRL

Bit	Access	INI	Symbol	Description
7:4	R/W		SD_CMD_DLY_SEL	Used to control the delay of SD_CMD
3:0				RSVD

14.3.27 SD_DATA01_TIMING_CTRL

- Name: SD_DATA01_TIMING_CTRL Register
- Width: 8bit
- Initial Value: 0x0

REG 14-36 (Offset 00DDh) SD_DATA01_TIMING_CTRL

Bit	Access	INI	Symbol	Description
7:4	R/W		SD_DATA1_DLY_SEL	Used to control the delay of SD_DATA1
3:0	R/W		SD_DATA0_DLY_SEL	Used to control the delay of SD_DATA0

14.3.28 SD_DATA23_TIMING_CTRL

- Name: SD_DATA23_TIMING_CTRL Register
- Width: 8bit
- Initial Value: 0x0

REG 14-37 (Offset 00DEh) SD_DATA23_TIMING_CTRL

Bit	Access	INI	Symbol	Description
7:4	R/W		SD_DATA3_DLY_SEL	Used to control the delay of SD_DATA3
3:0	R/W		SD_DATA2_DLY_SEL	Used to control the delay of SD_DATA2

14.4 SPDIO Control Registers

14.4.1 SPDIO_TXBD_ADDR

- Name: SPDIO_TXBD_ADDR Register
- Width: 32bit
- Initial Value: 0x0

REG 14-38 (Offset 01A0h) SPDIO_TXBD_ADDR

Bit	Access	INI	Symbol	Description
31:0	CPU: R/W DRV: R		SPDIO_TXFIFO_ADDR	The base address of TXBD The value of this register should be configured by CPU It should be 4 byte alignment (Because TXBD Size is 8 Byte).

14.4.2 SPDIO_TXBD_NUM

- Name: SPDIO_TXBD_NUM Register
- Width: 16bit
- Initial Value: 0x0

REG 14-39 (Offset 01A4h) SPDIO_TXBD_NUM

Bit	Access	INI	Symbol	Description
15:0	CPU: R/W DRV: R		SPDIO_TXBD_NUM	The Number of TXBD. Unit: Number The value of this register should be configured by CPU

14.4.3 SPDIO_TXBD_WPTR

- Name: SPDIO_TXBD_WPTR Register
- Width: 16bit
- Initial Value: 0x0

REG 14-40 (Offset 01A8h) SPDIO_TXBD_WPTR

Bit	Access	INI	Symbol	Description
15:0	R		SPDIO_TXBD_H2C_WPTR	When the packet has been sent from Host to TX Buffer, this index should be updated by SDIO IP.

14.4.4 SPDIO_TXBD_RPTR

- Name: SPDIO_TXBD_RPTR Register
- Width: 16bit
- Initial Value: 0x0

REG 14-41 (Offset 01ACh) SPDIO_TXBD_RPTR

Bit	Access	INI	Symbol	Description
15:0	CPU: RW DRV: R		SPDIO_TXBD_H2C_RPTR	When the packet has been processed by CPU and moved to MAC TX FIFO, this index should be updated by CPU.

14.4.5 SPDIO_RXBD_ADDR

- Name: SPDIO_RXBD_ADDR Register
- Width: 32bit
- Initial Value: 0x0

REG 14-42 (Offset 01B0h) SPDIO_RXBD_ADDR

Bit	Access	INI	Symbol	Description
31:0	CPU: RW DRV: R		SPDIO_RXBD_ADDR	The base address of RX BD. The value of this register should be configured by CPU. It should be 8-Byte alignment

14.4.6 SPDIO_RXBD_NUM

- Name: SPDIO_RXBD_NUM Register
- Width: 16bit
- Initial Value: 0x0100

REG 14-43 (Offset 01B4h) SPDIO_RXBD_NUM

Bit	Access	INI	Symbol	Description
15:0	CPU: RW DRV: R		SPDIO_RXBD_NUM	The total count of RX BD The value of this register should be configured by CPU. The unit is BD count

14.4.7 SPDIO_RXBD_C2H_WPTR

- Name: SPDIO_RXBD_C2H_WPTR Register
- Width: 16bit
- Initial Value: 0x0

REG 14-44 (Offset 01B6h) SPDIO_RXBD_C2H_WPTR

Bit	Access	INI	Symbol	Description
15:0	CPU: RW DRV: R		SPDIO_RXBD_C2H_WPTR	When the packet has been processed by CPU and to be moved to host, this index should be updated by CPU

14.4.8 SPDIO_RXBD_C2H_RPTR

- Name: SPDIO_RXBD_C2H_RPTR Register
- Width: 16bit
- Initial Value: 0x0

REG 14-45 (Offset 01B8h) SPDIO_RXBD_C2H_RPTR

Bit	Access	INI	Symbol	Description
15:0	CPU: R DRV: R		SPDIO_RXBD_C2H_RPTR	When the packet has been moved to host, this index should be updated by SDIO IP.

14.4.9 HCI_RX_REQ

- Name: HCI_RX_REQ Register
- Width: 8bit
- Initial Value: 0x0

REG 14-46 (Offset 01BAh) HCI_RX_REQ

Bit	Access	INI	Symbol	Description
7:1				
0	CPU: RW DRV: R		HCI_RX_REQ	CPU triggers this bit to enable SDIO IP RX transfer by fetch BD info. SDIO fetch BD to get the RX length and address and transfer RX packet to Host, and clear this bit when all BD transfer done.

14.4.10 CPU_RST_SDIO_DMA

- Name: CPU_RST_SDIO_DMA Register
- Width: 8bit
- Initial Value: 0x0

REG 14-47 (Offset 01BBh) CPU_RST_SDIO_DMA

Bit	Access	INI	Symbol	Description
7	CPU: RW DRV: R		CPU_RST_SDIO_DMA	CPU set this bit to reset SDIO DMA. This bit is auto clear.
6:1				RSVD
0	CPU: RW DRV: R		SDIO_DAT_EDGE_INV	Invert SDIO Latch input data edge.

14.4.11 SPDIO_RX_REQ_ADDR

- Name: SPDIO_RX_REQ_ADDR Register
- Width: 32bit
- Initial Value: 0x0

REG 14-48 (Offset 01BCh) SDIO_RX_REQ_ADDR

Bit	Access	INI	Symbol	Description
31:0	CPU: RW		SPDIO_RX_REQ_ADDR	The address of the requested RX Packet. This is unnecessary for driver.

	DRV: R		
--	--------	--	--

14.4.12 SPDIO_CPU_INT_MASK

- Name: SPDIO_CPU_INT_MASK Register
- Width: 16bit
- Initial Value: 0x0

REG 14-49 (Offset 01C0h) SPDIO_CPU_INT_MASK

Bit	Access	INI	Symbol	Description
15:12				RSVD
11	CPU: RW DRV: R		HOST_CMD11_INT_MSK	This bit is clear, and HOST_CMD11_INT is masked
10	CPU: RW DRV: R		HOST_WAKE_CPU_INT_MSK	This bit is clear, and HOST_WAKE_CPU_INT is masked
9	CPU: RW DRV: R		RX_BD_AVAI_INT_MSK	This bit is clear, and RX_BD_AVAI_INT is masked
8	CPU: RW DRV: R		RX_BD_FLAG_ERR_INT_MSK	This bit is clear, and RX_BD_FLAG_ERR_INT is masked
7	CPU: RW DRV: R		SDIO_RST_CMD_INT_MSK	This bit is clear, and SDIO_RST_CMD_INT is masked
6	CPU: RW DRV: R		RPWM2_INT_MSK	This bit is clear, and RPWM2_INT is masked
5	CPU: RW DRV: R		RPWM_INT_MSK	This bit is clear, and RPWM_INT is masked
4	CPU: RW DRV: R		H2C_MSG_INT_MSK	This bit is clear, and H2C_MSG_INT is masked
3	CPU: RW DRV: R		C2H_DMA_OK_MSK	This bit is clear, and C2H_DMA_OK_INT is masked
2	CPU: RW DRV: R		H2C_DMA_OK_MSK	This bit is clear, and H2C_DMA_OK_INT is masked
1	CPU: RW DRV: R		H2C_BUS_RES_FAIL_MSK	This bit is clear, and H2C_BUS_RES_FAIL_INT is masked
0	CPU: RW DRV: R		TXBD_H2C_OVF_MSK	This bit is clear, and TXBD_H2C_OVF is masked

14.4.13 SPDIO_CPU_INT_REG

- Name: SPDIO_CPU_INT_REG Register
- Width: 16bit
- Initial Value: 0x0

REG 14-50 (Offset 01C2h) SPDIO_CPU_INT_REG

Bit	Access	INI	Symbol	Description
15:12				RSVD
11	CPU: RW DRV: R		HOST_CMD11_INT	When Host sends CMD11 to notify that the signal voltage level is going to switch to 1.8v, the HW should issue this interrupt to notify the local CPU to do the LDO power voltage level switch.
10	CPU: R/W DRV: R		HOST_WAKE_CPU_INT	When Host Send TRX CMD53 while CPU is not ready (SYSTEM_CPU_RDY_IND=0, or CPU_RDY_IND=0), trigger this interrupt to wake CPU, and then indicate BUSY status to host. CPU should be able to receive this interrupt even CPU clock is gated; therefore, this interrupt should be connected to System On circuit.
9	CPU: R/W DRV: R		RX_BD_AVAI_INT	If the free RXBD Number become larger than FREE_RXBD_COUNT(0x1D8), trigger this interrupt. This interrupt trigger only once when free RXBD number cross FREE_RXBD_COUNT
8	CPU: R/W DRV: R		RX_BD_FLAG_ERR_INT	Trigger by SDIO to CPU when detect RX_BD error
7	CPU: R/W DRV: R		SDIO_RST_CMD_INT	Trigger by SDIO to CPU when SDIO is reset. CPU should be able to receive this interrupt even CPU clock is gated; therefore, this interrupt should be connected to System On circuit.
6	CPU: R/W DRV: R		RPWM2_INT	Trigger by SDIO to CPU that RPWM2 occurs (HRPWM is toggled) CPU should be able to receive this interrupt even CPU clock is gated; therefore, this interrupt should be connected to System On circuit.
5	CPU: R/W DRV: R		RPWM1_INT	Trigger by SDIO to CPU that RPWM1 occurs(HRPWM2 is written) CPU should be able to receive this interrupt even CPU clock is gated; therefore, this interrupt should be connected to System On circuit.
4	CPU: R/W DRV: R		H2C_MSG_INT	Trigger by SDIO to CPU that H2C_MSG_INT occurs(REG_SDIO_H2C_MSG is written)
3	CPU: R/W DRV: R		C2H_DMA_OK	Trigger by SDIO to CPU that packet is sent from RXFIFO to Host
2	CPU: R/W DRV: R		H2C_DMA_OK	Trigger by SDIO to CPU that packet is sent from Host to TXFIFO
1	CPU: R/W DRV: R		H2C_BUS_RES_FAIL	If SDIO is going to move packet to TX FIFO and fails to get bus resource, this interrupt raise. Driver Write 1 clear
0	CPU:		TXBD_H2C_OVF	If there is not enough TX_BD for TX Packet, this bit raise. Driver

	R/W DRV: R			Write 1 clear
--	---------------	--	--	---------------

14.4.14 CCPWM

- Name: CCPWM Register
- Width: 8bit
- Initial Value: 0x0

REG 14-51 (Offset 01C4h) CCPWM

Bit	Access	INI	Symbol	Description
7	CPU: R/W DRV: R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from “1” to “0” or from “0” to “1”.
6:4	CPU: R/W DRV: R		NO USED	NO USED
3	CPU: R/W DRV: R		WWLAN	1: Wake On WLAN State; 0: Normal State;
2	CPU: R/W DRV: R		RPS_ST	1: AP Register Active State; 0: AP Register Sleep State;
1	CPU: R/W DRV: R		WLAN_TRX	1: WLAN On; 0: WLAN Off.
0	CPU: R/W DRV: R		RSVD	

14.4.15 SYS_CPU_INDICATION

- Name: SYS_CPU_INDICATION Register
- Width: 8bit
- Initial Value: 0x0

REG 14-52 (Offset 01C5h) SYS_CPU_INDICATION

Bit	Access	INI	Symbol	Description
7:1	RSVD			
0	R/W		SYSTEM_CPU_RDY_IND	Used to indicate that CPU is ready for TRX. Default: 0 This bit is synchronized to CPU_RDY_IND(Offset 0x87)

14.4.16 CCPWM2

- Name: CCPWM2 Register
- Width: 16bit
- Initial Value: 0x0

REG 14-53 (Offset 01C6h) CCPWM2

Bit	Access	INI	Symbol	Description
15	CPU: R/W DRV: R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
14:0	CPU: R/W DRV: R		NO USED	NO USED

14.4.17 REG_CPU_H2C_MSG

- Name: REG_CPU_H2C_MSG Register
- Width: 32bit
- Initial Value: 0x0

REG 14-54 (Offset 01C8h) REG_CPU_H2C_MSG

Bit	Access	INI	Symbol	Description
31	CPU: R DRV: R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
30:0	CPU: R DRV: R		REG_CPU_H2C_MSG	Driver to FW Message

14.4.18 REG_CPU_C2H_MSG

- Name: REG_CPU_C2H_MSG Register
- Width: 32bit
- Initial Value: 0x0

REG 14-55 (Offset 01CCh) REG_CPU_C2H_MSG

Bit	Access	INI	Symbol	Description
31	CPU: R/W DRV: R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".

30:0	CPU: R/W DRV: R		REG_CPU_C2H_MSG	FW to Driver Message
------	-----------------------	--	-----------------	----------------------

14.4.19 CRPWM

- Name: CRPWM Register
- Width: 8bit
- Initial Value: 0x7

REG 14-56 (Offset 01D0h) CRPWM

Bit	Access	INI	Symbol	Description
7	CPU: R DRV: R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
6:4	CPU: R DRV: R		NO USED	NO USED
3	CPU: R DRV: R		WWLAN	1: Wake On WLAN State; 0: Normal State;
2	CPU: R DRV: R		RPS_ST	1: AP Register Active State; 0: AP Register Sleep State;
1	CPU: R DRV: R		WLAN_TRX	1: WLAN On; 0: WLAN Off.
0	CPU: R DRV: R			RSVD

14.4.20 CRPWM2

- Name: CRPWM2 Register
- Width: 16bit
- Initial Value: 0x0

REG 14-57 (Offset 01D2h) CRPWM2

Bit	Access	INI	Symbol	Description
15	CPU: R DRV: R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
14:0	CPU: R DRV: R		NO USED	NO USED

14.4.21 AHB_DMA_CTRL

- Name: AHB_DMA_CTRL Register
- Width: 32bit
- Initial Value: 0x74004040

REG 14-58 (Offset 01D4h) AHB_DMA_CTRL

Bit	Access	INI	Symbol	Description
31	CPU: RW DRV: R		DISPATCH_TXAGG_PKT	Enable SPDIO to dispatch the Aggregated TX packet.
30:28	CPU: RW DRV: R		AHB_BURST_TYPE	This field is used to control to SPDIO AHB support Burst type. 3'b100: Support Burst 16 DW 3'b010: Support Burst 8 DW 3'b001: Support Burst 4 DW Default is 3'b111, all support.
27:24	CPU: RW DRV: R		RX_AHB_BUSY_WAIT_CNT	When SPDIO RX transfer, AHB controller will wait BUSY counter for AHB access target not response READY signal. If timeout, AHB controller will issue AHB_BUS_RES_FAIL INT to CPU.
23:22	CPU: R DRV: R		AHBM_SPDIO_TRANS	SPDIO AHB Master HTRANS signal
21	CPU: R DRV: R		AHBM_SPDIO_READY	SPDIO AHB Master Hready signal
20	CPU: R DRV: R		NO USED	NO USED
19:16	CPU: R DRV: R		AHB_DMA_CS	AHB DMA state
15	CPU: R DRV: R		NO USED	NO USED
14:8	CPU: RW DRV: R		SPDIO_RXFF_WLEVEL	SPDIO FIFO RX water level Range 1~126, it can be modified only when AHB_DMA_CS = 000
7	CPU: R DRV: R		NO USED	NO USED
6:0	CPU: RW DRV: R		SPDIO_TXFF_WLEVEL	SPDIO FIFO TX water level Range 1~126, it can be modified only when AHB_DMA_CS = 000

14.4.22 FREE_RXBD_COUNT

- Name: FREE_RXBD_COUNT Register
- Width: 32bit
- Initial Value: 0x10100

REG 14-59 (Offset 01D8h) FREE_RXBD_COUNT

Bit	Access	INI	Symbol	Description
31:16	CPU: RW DRV: R		FREE_RXBD_COUNT	If SPDIO_RXBD_C2H_RPTR is updated and the free RXBD Number is larger than FREE_RXBD_COUNT, trigger RX_BD_AVAI_INT interrupt.
15:8	CPU: RW DRV: R		TX_BUFF_UNIT_SIZE	The Size of each single TX Buffer which is addressed by TX_BD Unit: 64Byte Ex: 0x01=>64Byte 0x10=>1024Byte
7	CPU: R DRV: R		NO USED	NO USED
6:0	CPU: R DRV: R		FIFO_CNT	The SPDIO Local FIFO counter, for debug usage

14.4.23 REG_CPU_H2C_MSG_EXT

- Name: REG_CPU_H2C_MSG_EXT Register
- Width: 32bit
- Initial Value: 0x0

REG 14-60 (Offset 01DCh) REG_CPU_H2C_MSG_EXT

Bit	Access	INI	Symbol	Description
31	R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
30:0	R		REG_CPU_H2C_MSG_EXT	Driver to FW Message

14.4.24 REG_CPU_C2H_MSG_EXT

- Name: REG_CPU_C2H_MSG_EXT Register
- Width: 32bit
- Initial Value: 0x0

REG 14-61 (Offset 01E0h) REG_CPU_C2H_MSG_EXT

Bit	Access	INI	Symbol	Description
31	R/W		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
30:0	R/W		REG_CPU_C2H_MSG_EXT	FW to driver Message

14.5 GSPI Control Registers

14.5.1 SPI_INT

- Name: SPI_INT Register
- Width: 32bit
- Initial Value: 0x0C000000

REG 14-62 (Offset 004h) SPI_INT

Bit	Access	INI	Symbol	Description
30:16	R/W		SPI_32us_INT_TIMEOUT	It is SPI INT timeout. The unit is 32usec. The default value is 0xc00 means that timeout is 98304usec (32*0xc00)
15:1	R		0	RSVD
0	R/W		R_EN_MASK_TIMER	The bit is only valid when EN_RXDMA_MASK_INT is 1. If this bit set to 0, timeout is disable. If this bit set to 1, it enable interrupt timeout.

14.5.2 SPI_HIMRxx

- Name: SPI_HIMR Register
- Width: 32bit
- Initial Value: 0xF

REG 14-63 (Offset 014h) SPI_HIMR

Bit	Access	INI	Symbol	Description
31:20	R/W		-	NO USED
19	R/W		CPWM2_MSK	This bit is clear, and CPWM2_INT is masked.
18	R/W		CPWM1_MSK	This bit is clear, and CPWM1_INT is masked.
17	R/W		C2H_MSG_MSK	This bit is clear, and C2H_MSG_INT is masked
16:5			-	
4	R/W		TXBD_OVERFLOW_MSK	This bit is clear, and TXBD_OVERFLOW is masked
3	R/W		TXAGG_SIZE_MISMATCH_MSK	This bit is clear, and TXAGG_SIZE_MISMATCH is masked
2	R/W		TXPKT_SIZE_OVER_BUFF_MSK	This bit is clear, and TXPKT_SIZE_OVER_BUFF is masked
1	R/W		AVAL_MSK	This bit is clear, and AVAL_INT INT is masked
0	R/W		RX_REQUEST_MSK	This bit is clear, and RX_REQUEST_INT is masked.

14.5.3 SPI_HISR

- Name: SPI_HISR Register
- Width: 32bit

- Initial Value: 0x0

REG 14-64 (Offset 018h) SPI_HISR

Bit	Access	INI	Symbol	Description
31:20	-		-	NO USED
19	R/W1C		CPWM2_INT	CPWM2 Write Interrupt: This interrupt will be raised when FW writes CPWM2_TOGGLING of CPWM2 register. Write 1 clear
18	R/W1C		CPWM1_INT	CPWM1 Write Interrupt: This interrupt will be raised when FW writes CPWM1_TOGGLING of CPWM1 register. Write 1 clear
17	R/W1C		C2H_MSG_INT	CPU to Host Message INT Status: This interrupt will be raised when FW sends a C2H MSG Write 1 clear
16:5	-		-	NO USED
4	R/W1C		TXBD_OVF	When host send TX packet and there is no enough TXBD, trigger this interrupt
3	R/W1C		TX_AGG_SIZE_MISMATCH	If the total size of the tx aggregation packets is LARGER the CMD53 Data Size, this interrupt trigger
2	R/W1C		TXPKT_SIZE_OVER_BUFF	If the Packet size (Aggregated or not) is larger than the BufferSize assigned in the TX_BD, this interrupt trigger
1	R		AVAL_INT	When the free page is lower than SPI_AVAIL_SPACE_L and higher than SPI_AVAIL_SPACE_H consequently, this interrupt occurs. When Host read FREE_TX_SPACE, the bit becomes zero.
0	DRV; R		RX_REQUEST	This bit is set when one complete RX aggregation packet is received into RXPKTBUF and RX_REQUEST_MSK is set to 1. Only when RX FIFO is empty, the bit becomes zero.

14.5.4 RX0_REQ_LEN

- Name: RX0_REQ_LEN Register
- Width: 32bit
- Initial Value: 0x40000000

REG 14-65 (Offset 01Ch) RX0_REQ_LEN

Bit	Access	INI	Symbol	Description
31	R		RX_REQ_LEN_RDY	Indicate RX0_REQ_LEN is ready
30	R		TXDMA_FIFO_OK	This bit is usage for GSPI mode, for GSPI need polling this bit to confirm previous TX transaction is from FIFO to SRAM and then start next NEW transaction. (could assign TXFIFO_WLEVEL to 1 for avoid polling bit)
29:24	-		-	RSVD
23:0	R		RX0_REQ_LEN	The requested RX Packet length; Unit: Byte For the RX Aggregation case, the length is total length of the packet. Each packet is 8-bytes aligned.

14.5.5 FREE_TX_BD_NUM

- Name: FREE_TX_BD_NUM Register
- Width: 32bit
- Initial Value: 0x0

REG 14-66 (Offset 020h) FREE_TX_BD_NUM

Bit	Access	INI	Symbol	Description
31:0	R		TXFF_FREE_BD	The free BD for TX_FIFO; unit: Number of Buffer Descriptor

14.5.6 TX_SEQNUM

- Name: TX_SEQNUM Register
- Width: 8bit
- Initial Value: 0x0

REG 14-67 (Offset 024h) TX_SEQNUM

Bit	Access	INI	Symbol	Description
7:0	R		TX_SEQNUM	Not used

14.5.7 HCPWM

- Name: HCPWM Register (Host domain, sync from CCPWM)
- Width: 8bit
 - Initial Value: 0x0

REG 14-68 (Offset 038h) HCPWM

Bit	Access	INI	Symbol	Description
7	R		TOGLLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
6:4	-		-	RSVD
3	R		WWLAN	1: Wake On WLAN State; 0: Normal State;
2	R		RPS_ST	1: AP Register Active State; 0: AP Register Sleep State;
1	R		WLAN_TRX	1: WLAN On; 0: WLAN Off.

14.5.8 HCPWM2

- Name: HCPWM2 Register (Host domain, sync from CCPWM2)
- Width: 16bit
- Initial Value: 0x0

REG 14-69 (Offset 03Ah) HCPWM2

Bit	Access	INI	Symbol	Description
15	R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
14:0	R		RSVD	Not defined.

14.5.9 SPI_AVAI_PGTH_L

- Name: SPI_AVAI_PGTH_L Register
- Width: 32bit
- Initial Value: 0x0

REG 14-70 (Offset 040h) SPI_AVAI_PGTH_L

Bit	Access	INI	Symbol	Description
31:0	R/W		SPI_AVAI_PGTH_L	Low threshold for AVAL_INT

14.5.10 SPI_AVAI_PGTH_H

- Name: SPI_AVAI_PGTH_H Register
- Width: 32bit
- Initial Value: 0x0

REG 14-71 (Offset 044h) SPI_AVAI_PGTH_H

Bit	Access	INI	Symbol	Description
31:0	R/W		SPI_AVAI_PGTH_H	High threshold for AVAL_INT

14.5.11 SPI_RX_AGG

- Name: SPI_RX_AGG Register
- Width: 32bit

- Initial Value: 0x0

REG 14-72 (Offset 048h) SPI_RX_AGG

Bit	Access	INI	Symbol	Description
31:16	R/W		RSVD	
15	R/W		SPI_RX_AGG_EN	The SPI RX Aggregation Enable bit
14:8	R/W		SPI_RX_AGG_BDTO	The SPI RX Aggregation Timeout value; Timeout unit is 32usec @ 100Mhz sys_clk
7:0	R/W		SPI_RX_AGG_BDCNT	The SPI RX Aggregation BD Count Number

14.5.12 REG_SPI_H2C_MSG

- Name: REG_SPI_H2C_MSG Register (Host domain, sync to REG_CPU_H2C_MSG)
- Width: 32bit
- Initial Value: 0x0

REG 14-73 (Offset 04Ch) REG_SPI_H2C_MSG

Bit	Access	INI	Symbol	Description
31	R/W		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
30:0	R/W		REG_SPI_H2C_MSG	Driver to FW Message

14.5.13 REG_SPI_C2H_MSG

- Name: REG_SPI_C2H_MSG Register (Host domain, sync from REG_CPU_C2H_MSG)
- Width: 32bit
- Initial Value: 0x0

REG 14-74 (Offset 050h) REG_SPI_C2H_MSG

Bit	Access	INI	Symbol	Description
31	R		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
30:0	R		REG_SPI_C2H_MSG	FW to Driver Message

14.5.14 HRPWM

- Name: HRPWM Register (Driver to F/W, Host domain, sync to CRPWM)
- Width: 8bit
- Initial Value: 0x0

REG 14-75 (Offset 080h) HRPWM

Bit	Access	INI	Symbol	Description
7	R/W		TOGGLING	Toggling Bit: this is the one bit sequence number field. Interrupt is issued when this bit is changed from "1" to "0" or from "0" to "1".
6:4	-		-	RSVD
3	R/W		WWLAN	1: Wake On WLAN State; 0: Normal State;
2	R/W		RPS_ST	1: AP Register Active State; 0: AP Register Sleep State;
1	R/W		WLAN_TRX	1: WLAN On; 0: WLAN Off.

14.5.15 HPS_CLKR

- Name: HPS_CLKR Register
- Width: 8bit
- Initial Value: 0x0

REG 14-76 (Offset 084h) HPS_CLKR

Bit	Access	INI	Symbol	Description
7:0	RW		NO USED	NO USED

14.5.16 CPU_INDICATION

- Name: CPU_INDICATION Register
- Width: 8bit
- Initial Value: 0x0

REG 14-77 (Offset 087h) CPU_INDICATION

Bit	Access	INI	Symbol	Description
7:4	RW		SDIO_TEST_BUSY_CNT	For Device test BUSY control count, BUSY time $\sim=$ BUSY_CNT * 16-SD_CLK
3:1	-		-	RSVD

0	R		CPU_RDY_IND	Used to indicate that CPU is ready for TRX. Default: 0 This bit is synchronized from SYSTEM_CPU_RDY_IND(Offset 0x1C5)
---	---	--	-------------	--

14.5.17 32K_TRANSPARENT

- Name: 32K_TRANSPARENT Register
- Width: 8bit
- Initial Value: 0x0

REG 14-78 (Offset 088h) 32K_TRANSPARENT

Bit	Access	INI	Symbol	Description
7:1	-		-	
0	R/W		EN32K_TRANS	Enable 32K transparent. Only available for SDIO interface. Enable the FW can enter/exit its power saving by itself. The driver no needs to control the FW power mode. The SDIO interface will report 'busy' state when FW is in power saving state.

14.5.18 32K_IDLE

- Name: 32K_IDLE Register
- Width: 16bit
- Initial Value: 0x2EE0

REG 14-79 (Offset 08Ah) 32K_IDLE

Bit	Access	INI	Symbol	Description
15:0	R/W		EN32K_TRANS_IDLE_TIME	32K transparent Idle time

14.5.19 DELY_LINE_SEL

- Name: DELY_LINE_SEL Register
- Width: 8bit
- Initial Value: 0x0

REG 14-80 (Offset 08Ch) DELY_LINE_SEL

Bit	Access	INI	Symbol	Description
7:4	--		--	
3:0	R/W		DELY_LINE_SEL	Delay line selection, Unit: ns

			Change this value will delay the SPI_MISO transient time.
--	--	--	---

14.5.20 SPI_CFG

- Name: SPI_CFG Register
- Width: 8bit
- Initial Value: 0x3

REG 14-81 (Offset 0F0h) SPI_CFG

Bit	Access	INI	Symbol	Description
7:2	--		--	
1	R/W		SPI_ENDIAN	0:big endian 1: little endian (default)
0	R/W		SPI_WORD_LEN	0:16 bits word length 1:32 bits word length (default)

15 SPI NOR Flash Controller (SPIC)

15.1 Overview

15.1.1 Application Scenario

The SPI NOR Flash Controller (SPIC) is a peripheral to control SPI-NOR flash. SPI-NOR flash is a memory device to storage data or codes for systems with limited space, pins and power. With the help of SPIC, user can erase or program data into flash memory. Load data from flash memory when system requests for data. Furthermore, current flash memory supports higher data rate with multiple IO modes which makes execute cod directly from flash (XIP) possible. Users stores application codes that is less sensitive to timing requirement and the processor fetches instructions to execute. The processor no longer has to load these instructions from internal RAM before executing.

15.1.2 Features

- It is a synchronous design which supports AXI bus interface
 - 32 bit data width
 - Byte enable write is not supported. Register access is always 4 bytes.
- Support multiple IO modes for various flash types. Please refer to available lists to check qualified flash part numbers.
 - One IO
 - Dual Output
 - Quad IO / QPI
- Maximum speed support for different IO modes is listed below
 - One IO: 25MHz
 - Multiple IO mode: 50MHz
- Allow data access through auto mode or user mode
 - Auto mode: Translate AXI memory-mapping read/write transfer into SPI flash access, a convenient way to access flash without setting many registers. When D-cache is enabled, the read performance in auto mode is better than user mode.
 - User mode: Haves much better performance to program a block of data. It is more flexible to send various commands to flash. However, more registers have to be configured manually.
- Support eXecute In Place (XIP) feature

- The processor is able to execute codes from flash directly.

15.1.3 Architecture

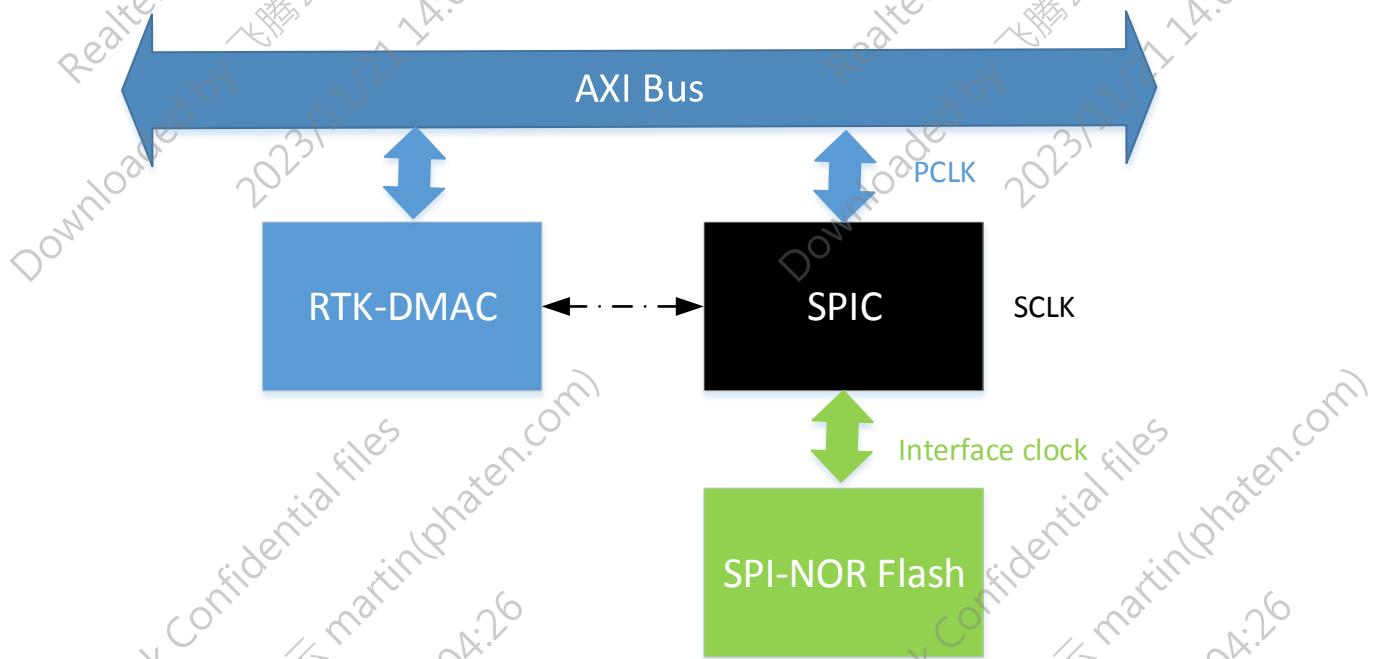


Figure 15-1 Ameba-Z II SPIC in A Complete System

Ameba-Z II's SPI flash controller (SPIC) is a slave device on AXI bus system. There are up to three clock domains inside SPIC. The first clock domain is PCLK, which is the AXI bus clock. Since SPIC is an AXI slave, it has AXI interface to communicate with AXI bus. The communication between SPIC and AXI bus operates at the speed of the AXI bus clock. The second clock domain is SCLK. This clock is the actual clock that SPIC registers reference. All the register controls including data push / pop are operated on this clock domain. The last clock domain, interface clock, is generated by a SPIC internal circuit. The flash device is operated at the speed of the interface clock. SCLK is selectable. It can either directly come from PCLK or from another clock source. If SCLK = PCLK, the number of clock domain is reduced to two. For Ameba-Z II system, the PCLK is 50 MHz while the SCLK is 200 MHz. The interface clock is configurable by the baud rate register. To achieve optimal throughput, the interface clock of one IO mode is set to 25 MHz and the multiple IO mode is accelerated to 50 MHz.

Flash memory can be accessed by the processor or DMA engine, RTK-DMAC, through SPIC. With the memory mapping transfer feature, flash memory acts like SRAM. SPIC takes care of all the transfer and automatically transforms bus commands to the corresponding flash commands.

15.1.3.1 Functional Block Diagram

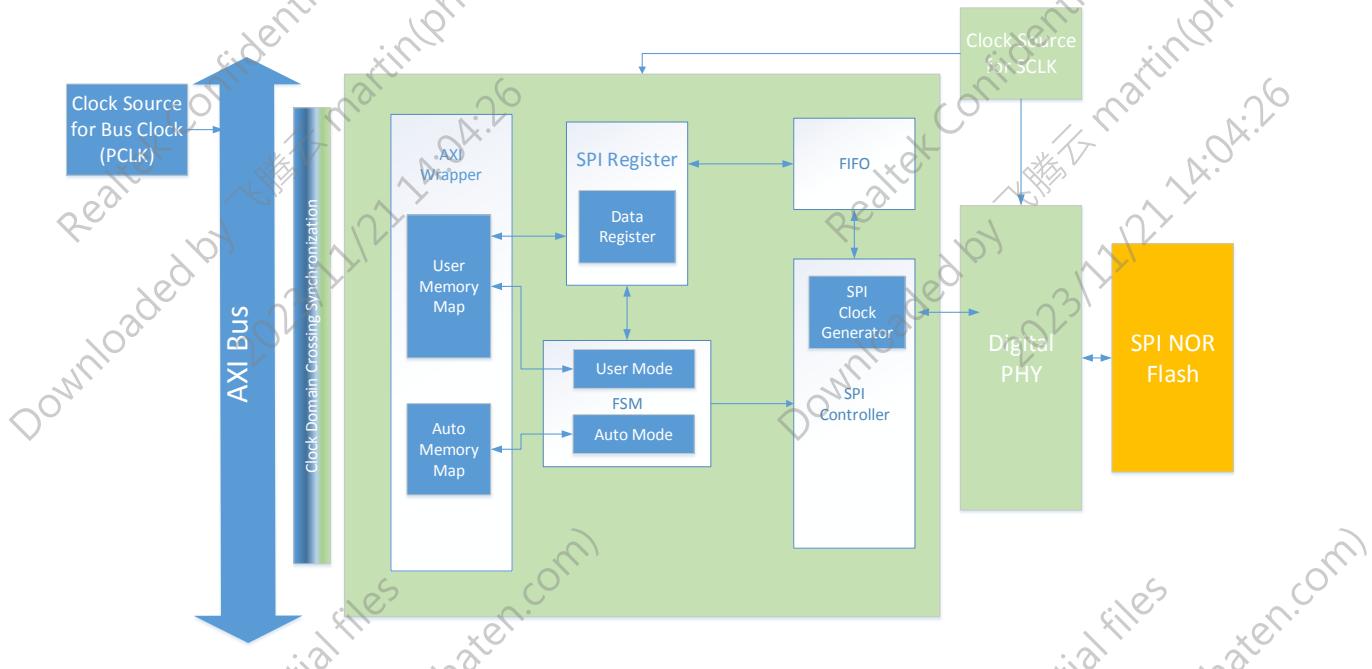


Figure 15-2 SPIC Block Diagram

- PCLK: The clock of the AXI bus.
- SCLK: The clock that SPIC operates.
- Clock Domain Crossing Synchronization: If SCLK is not equal to PCLK, we need to a synchronization circuit to synchronize between SPIC and AXI bus.
- AXI Wrapper: Communicate with AXI bus with AXI interface signals. It also decodes the address coming from AXI bus to determine user memory mapping or auto memory mapping.
 - User memory mapping: If the address received by AXI wrapper is the offset 0x4002xxxx, this command is direct to User Mode FSM. Users configure register values or even enable data transfer manually by using user memory mapping.
 - Auto memory mapping: If the address received by AXI wrapper is the offset 0x98xxxxxx, this command is directed to Auto Mode FSM. The AXI command is translated into the corresponding flash command automatically.
- FSM Control: Finite State Machine Control controls the state change of SPIC. User mode and Auto mode state machine are controlled separately.
- Digital PHY: A hardware to calibrate available windows when SPIC receives data from flash.
- SPI Controller: Take care of data transfer to flash. AXI commands are translated into corresponding flash commands by this hardware in auto mode. In user mode, it sends commands to the flash according to FIFO data and register values set by users.

- SPI Clock Generator: Generate interface clock spi_sclk to flash according to the value of SPIC_BAUDR register.

15.2 Functional Description

15.2.1 Overview

SPIC is a peripheral to control Serial NOR Flash device. The NOR flash serves as a storage device to store our RAM codes or user data. As mentioned in many flash data sheets, flash is able to support multiple IO modes to achieve higher data rate by using more data IO pins. In theory, SPIC can communicate with flash under any IO mode as long as the target flash can support such IO mode. The IO modes basically are categorized into two types. Legacy flash is accessed with one IO mode only. The most common used flash is QSPI/QPI flash, it can be accessed with 2IO pins or 4IO pins to significantly improve throughput in comparison with legacy flash. If QSPI/QPI flash is used, the processor is able to execute codes from flash directly which adds flexibility. This feature is especially useful for embedded systems which have very limited SRAM space.

15.2.2 SPIC Pin Connection

SPIC supports one IO mode, dual IO mode and quad IO mode. One IO mode uses only one IO (DI) to send data and one IO (DO) to receive data. spi_csn and spi_clk, which is the interface clock to the flash, are connected to CS and CLK pins respectively. Since SPIC and FLASH are pin-to-pin connected, we should not leave WP and HOLD pins floating if write protect and hold features are not enabled. These two pins are active low. In this case, WP and HOLD pins should be weakly pull high.

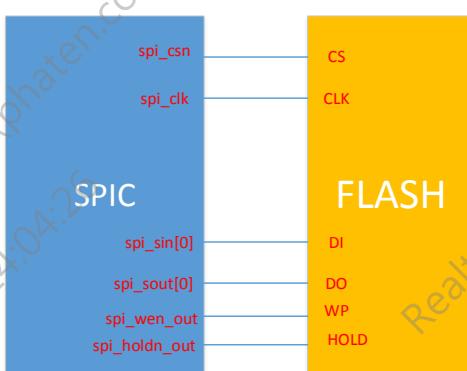


Figure 15-3 Single Channel Connection (for One IO Mode)

Under multiple IO mode, the multi-Channel connection is shown as below. Each IOx pin is connected to the corresponding spi_sin[x] / spi_sout[x] pin. Spi_sin[x] and spi_sout[x] shares the same pin of SPIC. SPIC will determine to switch between input mode or output mode depending on current operation.

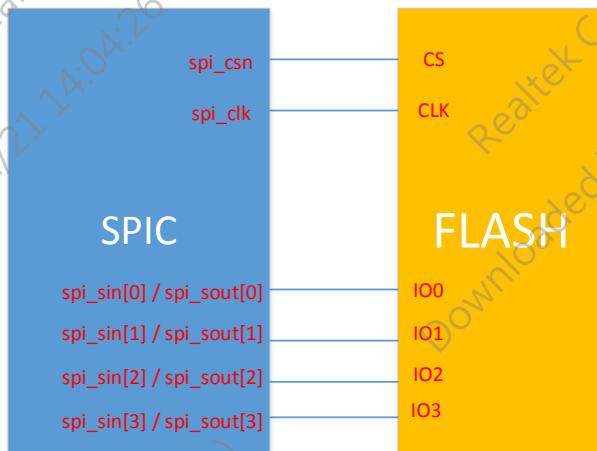


Figure 15-4 Multi-Channel Connection

15.2.3 SPIC Transfer Protocol

The flash controller adopts SPI interface to communicate with flash. We have four SPI formats to choose from in light of different combinations of serial clock phase (SCPH) and serial clock polarity (SCPOL) theoretically. However, most flash devices only support mode 0 & mode 3. To be compatible with all flash type, the default SPI mode we set for the flash controller is mode 0. Unless the target flash does not support mode 0, we suggest users do not alter the SPI Mode. The SPI mode is configured in the SPIC_CTRLR0 register.

- SCPOL = 0 (Inactive state of serial clock is low)
SCPH = 0 (Serial clock toggles in middle of first data bit)

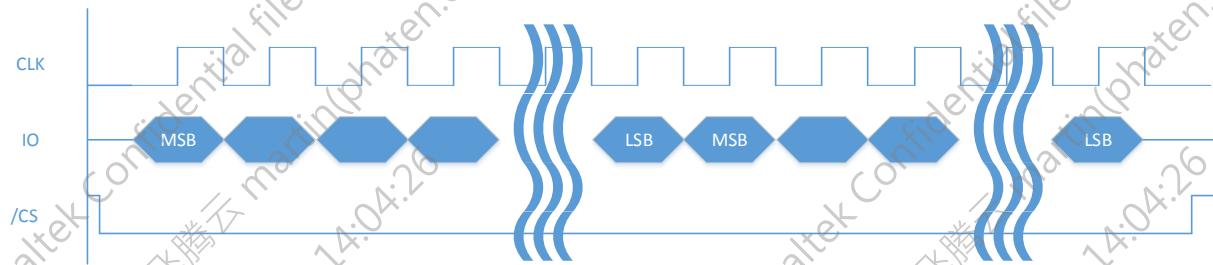


Figure 15-5 SPI Protocol: Mode 0

- SCPOL = 1 (Inactive state of serial clock is high)
SCPH = 1 (Serial clock toggles at start of first data bit)

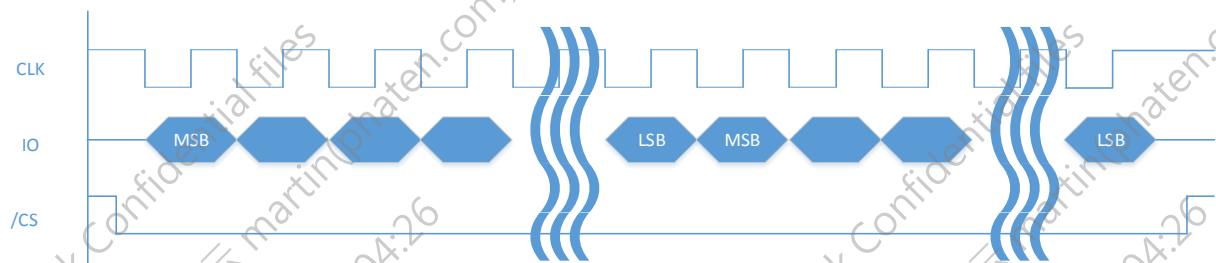


Figure 15-6 SPI Mode Protocol : Mode 3

15.2.4 SPIC Transfer Mode

Flash transfer can be divided into three phases, command phase, address phase and data phase. Read commands and write commands always contain these three phases. Data phase does not appear for erase commands. Erase commands may include address phase to specify the address to be erased. Read status register command or Write status register command may or may not have address phase but always has data phase. No matter what commands SPIC issues, command phase and address phase are always sent by SPIC. The direction of data phase is how we define the transfer mode under **user mode** for SPIC. If data is transferred by SPIC, the transfer mode is defined as transmit mode. If data comes from flash, the transfer mode is receive mode. The general format of SPIC transfer mode is shown as below. The data phase of Tx-data and Rx-data are present separately. The dummy cycle phase includes dummy cycles required for particular read commands defined in flash spec. If the certain phase is 0 byte, it indicates the phase does not exist.

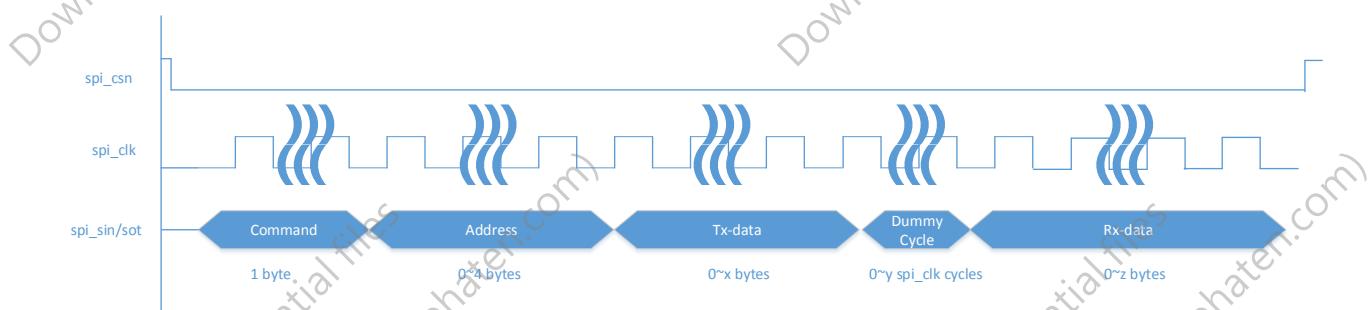


Figure 15-7 General Format of Transfer Mode

15.2.4.1 Transmit Mode

If no data comes from flash for data phase, we should set the transfer mode to transmit mode. The transmit mode is classified into three types. The first type is shown in the Figure 15-8. Only one command byte is sent.

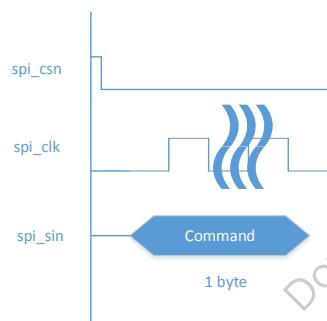


Figure 15-8 Transmit Mode: Type 1

The second type of transmit mode includes command phase followed by tx-data phase or address phase. This is useful for some erase commands or write status register commands.

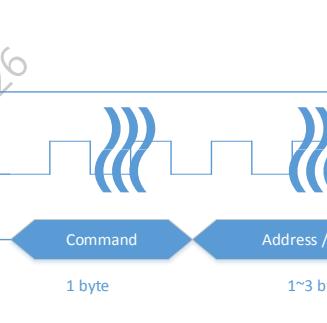


Figure 15-9 Transmit Mode: Type 2

The last type is the normal flash write sequence. It contains command phase, address phase and tx-data phase.

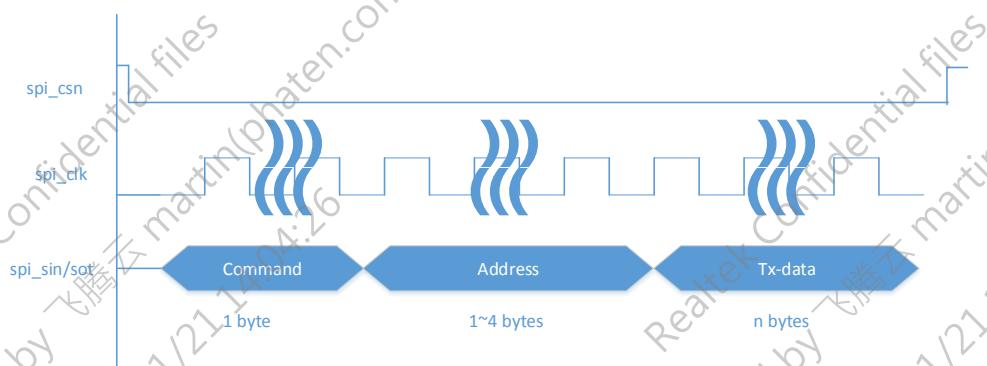


Figure 15-10 Transmit Mode: Type 3

15.2.4.2 Receive Mode

In receive mode, the spi_clk still toggles to read data from flash after the command and the address phase. The number of dummy cycles are defined in flash spec for different flash commands. The rx-data bytes are controlled by CTRLR1 register. SPIC counts the number of spi_clk depending on the value of CTRLR1. When cycles reach the value, SPIC de-asserts spi_csn to finish the transfer. The data read sequence in receive mode is shown as below.

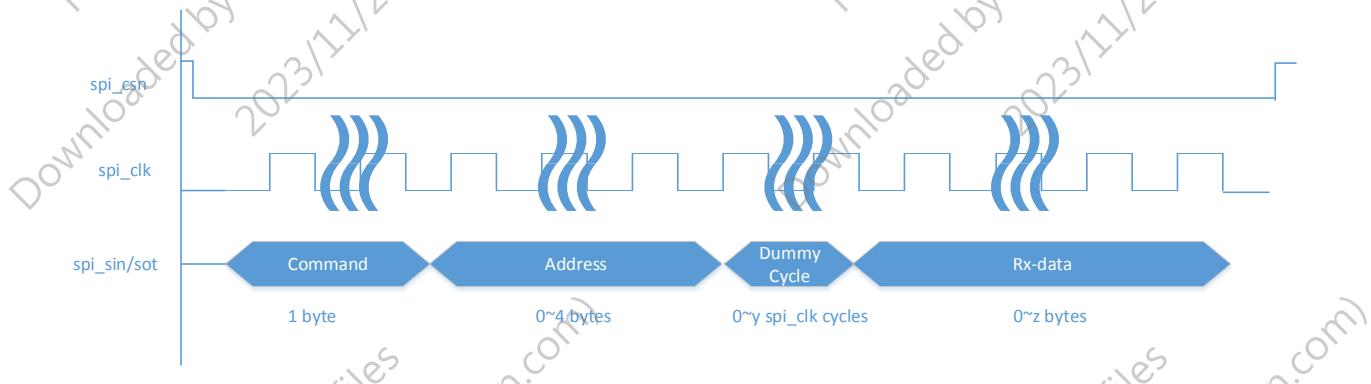


Figure 15-11 Receive Mode

15.2.5 SPIC Operation Mode

15.2.5.1 User Mode

User mode is a typical software flow to implement serial transfer and control SPIC registers. The registers access are communicated through user memory map with base memory address 0x4002xxxx. User mode operations are managed by User mode FSM. User mode provides flexibility to handle various flash commands since each flash command may need different transfer sequences. For instance, one IO read flash command uses one command/address/data channel while Dual IO flash read command uses one channel for command phase and two channels for address phase and data phase. A write command may carry up to 256 bytes data after address phase, but a chip erase command simply sends one byte command without any data. By controlling user mode registers, we are able to make different transfer sequences for various flash commands.

15.2.5.2 Auto Mode

Auto mode provides an intuitive way to read and write flash. Flash access behavior acts like SRAM access for users under auto mode. The data access is accomplished by auto memory map with base memory address 0x98xxxxxx. Auto mode operations are managed by Auto mode FSM. Most of times, users expect to read data from flash with a single step without configuring registers. Also, they wish SPIC can handle data access to flash by itself without user intervention when the processor or cache fetches data or instructions. To meet this demand, SPIC modularizes some flash read commands and flash write commands. We are going to discuss auto mode read and auto mode write in the following paragraphs.

Users can enable specific read command or write command in auto mode control register. A read command or write command received by SPIC is automatically translated into the corresponding flash commands and sends to flash. For examples, if user set Dual IO read command in the auto mode control register, any read command coming from the bus is translated into 0xBB read command, which is defined in flash datasheets, and sends to flash.

In auto mode read, data received from flash is handled automatically by SPIC. SPIC is allowed to forward partial data to the bus master when it collects certain amount of data in the FIFO before the whole transfer is done. This is not in case for user mode, data is read out of FIFO only when flash finishes the transfer. Therefore, if cache is enabled, auto mode read outperforms user mode read.

Unlike SRAM, flash memory takes some times to program data into flash array after SPIC sends write command. Auto mode write is supposed to be able to access just like SRAM when we enable auto write check function. For compatibility consideration, we do not encourage users to adopt auto mode write. After the flash controller issues program / erase / write status commands to flash, according to the spec of flash, we should wait a period until flash returns from busy state to ready state. This can be done by sending the read status register command.

However, the auto write check function cannot be compatible with all flash types. Some flash types use very different ways to check the flash status after sending program / erase / write status commands. The flash status register should be checked manually by user mode. The other reason we do not encourage to use auto mode program is the performance consideration. Since auto mode is directly translated the bus write command, the burst size is determined by the bus master. In Ameba-Z II system, the maximum burst size issued by the bus masters to access SPIC is cache. The size of each cache line is 32 bytes. In other words, each write command coming from the bus to SPIC only has 32 bytes burst size. A write command

programs 32 bytes data at a time into flash. In contrast to the program operation in user mode, we can program 256 bytes data at most with one write command. It is worthy to note that it takes a much longer time to wait for flash returning from busy status to ready status after a write or erase command. SPIC spends few microseconds to send commands to flash while flash requires few milliseconds to program data into flash array. Therefore, it is more efficient to carry as many data as possible with a single write command.

15.2.6 Flash Calibration Mechanism

SPI calibration is a process to optimize read timing of the SPIC. The procedure involves with two parameters: baud rate, read dummy cycle. Baud rate is directly associated with performance of SPIC. The smaller value of baud rate means the higher operating frequency of the SPIC. Read dummy cycle is necessary to ensure the flash has enough time to prepare data and make sure the data read window is valid. We may have to insert some dummy cycles to postpone data receiving to handle dummy periods induced by critical paths issue or intrinsic flash features after the last falling edge of the address phase (CMD -> Address -> Data). Usually additional dummy cycles are necessary when the flash is operating at high frequency or multiple IO modes.

Given the context in the last paragraph, the procedure of SPI calibration is implemented mainly by two “for” loops to determine the optimized parameters of baud rate and read dummy cycle. The first loop determines the baud rate while the second loop inside the first one decides dummy cycles. Once the baud rate is selected, the calibration is started from the smallest viable dummy cycles in the second loop. If the current dummy cycle can read back correct golden patterns, which is stored in the flash, this dummy cycle is marked as a valid dummy cycle, and the algorithm proceeds to the next loop. If several consecutive dummy cycles can all read correct golden patterns, we then pick the middle value of the available window as our final selection of the dummy cycle. Otherwise, the smallest available dummy cycle is picked. For example, if dummy cycle 2 ~ dummy cycle 10 all can read correct golden patterns, the value $(2+10) / 2 = 6$ is selected. Once at least one available window is found, the program breaks these two loops and records the result. The result is stored in the flash memory later so that we don't have to re-do calibration if system is rebooted.

15.3 Control Registers

15.3.1 SPIC_CTRLR0

- Name: Control Register 0
- Width: 32 bits
- Initial Value: 0x01000000

The CTRLR0 controls settings associated with user mode. It cannot be programmed when SSIENR is active.

REG 15-1 (Offset 0000h) SPIC_CTRLR0

Bits	Access	INI	Symbol	Description
23:31	R/W	0x0	RSVD	Internal Use
22	R/W	0x0	FAST_RD	Enable fast read command in user mode. 0: Disable 1: Enable
21:20	R/W	0x0	CMD_CH	Command channel number. 0: Single channel 1: Dual channel 2: Quad channel 3: Reserved
19:18	R/W	0x0	DATA_CH	Data channel number. 0: Single channel 1: Dual channel 2: Quad channel 3: Reserved
17:16	R/W	0x0	ADDR_CH	Address channel number. 0 : Single channel 1: Dual channel 2: Quad channel 3: Reserved
9:8	R/W	0x0	TMOD	Transfer mode. 0: Transmit mode 1,2,3: Receive mode
7	R/W	0x0	SCOPL	Serial Clock Polarity. Used to select the polarity of the inactive serial clock, which is held inactive when SPIC is not actively transferring data on the serial bus. 0 – Inactive state of serial clock is low 1 – Inactive state of serial clock is high
6	R/W	0x0	SCPH	Serial Clock Phase. The serial clock phase selects the relationship of the serial clock with the chip select signal. 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. 0: Serial clock toggles in middle of first data bit 1: Serial clock toggles at start of first data bit

15.3.2 SPIC_CTRLR1

- Name: Control Register 1
- Width: 12 bits
- Initial Value: 0x0000

It is used to count the number of data frames when data is received in user mode. If setting to 0, SPIC would not receive any data in user mode. It can't be programmed when SSIENR is active.

REG 15-2 (Offset 0004h) SPIC_CTRLR1

Bits	Access	INI	Symbol	Description
11:0	R/W	0x0	NDF	Indicate the number of data frames. When we implement a receive operation in user mode, SPIC would receive data continuously until the number of data frames is equal to NDF.

15.3.3 SPIC_SSIENR

- Name: SPIC Enable Register
- Width: 1 bit
- Initial Value: 0x0000

It is used to enable SPIC. If SSIENR is disabled, all transfers of user mode are halted. User can't program some control registers if SSIENR is enabled.

REG 15-3 (Offset 0008h) SPIC_SSIENR

Bits	Access	INI	Symbol	Description
0	R/W	0x0	SPIC_EN	Enable or disable SPIC. 1: Enable SPIC 0: Disable SPIC

15.3.4 SPIC_SER

- Name: Slave Enable Register
- Width: 1 bit
- Initial Value: 0x0000

It is used to select target SPI Flash to communication in user mode. The value is always equal to 1. It can't be programmed when SSIENR is active.

REG 15-4 (Offset 0010) SPIC_SER

Bits	Access	INI	Symbol	Description
0	R/W	0x0	SER	SPIC only has one slave select line. This bit should be always set.

15.3.5 SPIC_BAUDR

- Name: Baud Rate Select Register
- Width: 12 bits
- Initial Value: 0x0008

It is used to determine the interface clock, spi_sclk, to the flash memory. It is usually used in normal SPI command except fast read command. If BAUDR is setting 0 and a command is sent, SPIC would halt in IDLE state. It can't be programmed when SSIENR is active.

REG 15-5 (Offset 0014) SPIC_BAUDR

Bits	Access	INI	Symbol	Description
15:0	R/W	0x8	SCKDV	Define spi_sclk divider value. The frequency of spi_sclk is derived from: Frequency of spi_sclk = Frequency of spic_clk / (2*SCKDV).

15.3.6 SPIC_TXFLR

- Name: Transmit FIFO Level Register
- Width: 7 bits
- Initial Value: 0x0000

It is used to inform the number of valid data entries for normal transfer except receiving data.

REG 15-6 (Offset 0020h) SPIC_TXFLR

Bits	Access	INI	Symbol	Description
6:0	R	0x0	TXTFL	Transmit FIFO level. Indicate the FIFO entry level of valid data under transmit mode (TMOD = 0).

15.3.7 SPIC_RXFLR

- Name: Receive FIFO Level Register
- Width: 7 bits

- Initial Value: 0x0000

It is used to inform the number of valid data entries when the controller receives data.

REG 15-7 (Offset 0024h) SPIC_RXFLR

Bits	Access	INI	Symbol	Description
6:0	R	0x0	RXTFL	Receive FIFO level. Indicate the FIFO entries of valid data under receive mode (TMOD = 1, 2, 3).

15.3.8 SPIC_SR

- Name: Status Register
- Width: 7 bits
- Initial Value: 0x0006

It is used to inform the status during a transfer.

REG 15-8 (Offset 0028h) SPIC_SR

Bits	Access	INI	Symbol	Description
6	R	0x0	DCOL	Transmitting Status. This bit is set when SPIC is transmitting command, address or data to the data register. By observing this bit, users can avoid reading wrong data at the wrong time if data is not ready to be read.
5	R	0x0	TXE	Transmission error. Set if FIFO is empty and starting to transmit data to SPI Flash. This bit is cleared when read.
4	R	0x0	RFF	Receive FIFO full. Set if FIFO is full in receiving mode. This bit is cleared when read.
3	R	0x0	RFNE	Receive FIFO is not empty. Set if FIFO is not empty in receiving mode. This bit is cleared when read.
2	R	0x1	TFE	Transmit FIFO is empty. Set if FIFO is empty in transmit mode, else it is cleared when it has data in FIFO.
1	R	0x1	TFNF	Transmit FIFO is not full. Set if FIFO is not full in transmit mode, else it is cleared when FIFO is full.
0	R	0x0	BUSY	SPIC busy flag. Set if SPIC is still transmitting to or receiving data from SPI Flash.

15.3.9 SPIC_IDR

- Name: Identification Code Register
- Width: 32 bits
- Initial Value: 0x0203FF01

It is read only register to define peripheral identification code of SPIC.

REG 15-9 (Offset 0058h) SPIC_IDR

Bits	Access	INI	Symbol	Description
31:0	R	0x020 3FF01	IDCODE	Contain the decimal value of SPIC version

15.3.10 SPIC_VERSION

- Name: SPIC Version Register
- Width: 32 bits
- Initial Value: 0x01080119

It is read only register that stores the SPIC version number.

REG 15-10 (Offset 005Ch) SPIC_VERSION

Bits	Access	INI	Symbol	Description
31:0	R	0x010 8011 9	SPIC_VERSION	Current SPIC Version.

15.3.11 SPIC_DR

- Name: Data Register
- Width: 32 bits
- Initial Value: 0x0000

It is data buffer of FIFO for pushing/popping data. SPIC supports 8-bit byte, 16-bit (half word), 24-bit (tri-byte), and 32-bit (word) access. SPIC also supports burst read and the maximum length is 64 bytes to improve data transfer performance.

REG 15-11 (Offset 0060h) SPIC_DR

Bits	Access	INI	Symbol	Description
31:0	R/W	0x0	DR	It is a data buffer for 8-bit width FIFO. For example: If accessing in word data byte, SPIC would read/write 4 entries of FIFO.

15.3.12 SPIC_READ_FAST_SINGLE

- Name: Fast Read Command Register
- Width: 8 bits
- Initial Value: 0XB

It is used to implement fast read command in auto mode. It can't be programmed when SSIENR is active.

REG 15-12 (Offset 00E0h) SPIC_READ_FAST_SINGLE

Bits	Access	INI	Symbol	Description
7:0	R/W	0x0B	FRD_CMD	Indicate SPI Flash command value of fast read command. The baud rate is derived from FBAUDR instead of BAUDR.

15.3.13 SPIC_READ_DUAL_DATA

- Name: Dual Read Command Register
- Width: 8 bits
- Initial Value: 0x3B

It is used to implement dual data read command in auto mode. It can't be programmed when SSIENR is active.

REG 15-13 (Offset 00E4h) SPIC_READ_DUAL_DATA

Bits	Access	INI	Symbol	Description
7:0	R/W	0x3B	RD_DUAL_O_CM D	Indicate SPI Flash command value of dual data read command. (Dual output mode, only data phase is two IO)

15.3.14 SPIC_READ_DUAL_ADDR_DATA

- Name: Dual IO Read Command Register
- Width: 8 bits
- Initial Value: 0xBB

It is used to implement dual address and data read command in auto mode. It can't be programmed when SSIENR is active.

REG 15-14 (Offset 00E8h) SPIC_READ_DUAL_ADDR_DATA

Bits	Access	INI	Symbol	Description
7:0	R/W	0xBB	RD_DUAL_IO_CM D	Indicate SPI Flash command value of dual address and data read command. (Dual IO mode, data phase and address phase are two IO)

15.3.15 SPIC_READ_QUAD_DATA

- Name: Quad Read Command Register
- Width: 8 bits
- Initial Value: 0x6B

It is used to implement quad data read command in auto mode. It can't be programmed when SSIENR is active.

REG 15-15 (Offset 00ECh) SPIC_READ_QUAD_DATA

Bits	Access	INI	Symbol	Description
7:0	R/W	0x6B	RD_QUAD_O_CM D	Indicate SPI Flash command value of quad data read command. (Quad Output mode, data phase is four IO)

15.3.16 SPIC_READ_QUAD_ADDR_DATA

- Name: Quad IO Read Command Register
- Width: 24 bits
- Initial Value: 0xA500EB

It is used to implement quad address and data read command in auto mode. Some flash types support continuous read mode that can save command phase. Under the continuous read mode, a control parameter should be inserted right after address phase. This register can configure the control parameter for auto mode when flash enters continuous read mode. It can't be programmed when SSIENR is active.

REG 15-16 (Offset 00F0h) SPIC_READ_QUAD_ADDR_DATA

Bits	Access	INI	Symbol	Description
23:16	R/W	0xA5	PRM_VALUE	The control parameter of continuous read mode. 0xA5: The control parameter to enable continuous read mode 0x00: Exit continuous read mode
7:0	R/W	0xEB	RD_QUAD_IO_C MD	Indicate SPI Flash command value of quad address and data read command. (Quad IO mode, data phase and address phase are four IO)

15.3.17 SPIC_WRITE_SINGLE

- Name: Single IO Page Program Register
- Width: 8 bits
- Initial Value: 0x02

It is used to implement single address and data write command in auto mode. It can't be programmed when SSIENR is active.

REG 15-17 (Offset 00F4h) SPIC_WRITE_SINGLE

Bits	Access	INI	Symbol	Description
7:0	R/W	0x02	WR_CMD	Indicate SPI Flash command value of page program command.

15.3.18 SPIC_WRITE_ENABLE

- Name: Write Enable Command Register
- Width: 8 bits
- Initial Value: 0x06

It is used to implement write enable command before implementing write command of auto mode. It can't be programmed when SSIENR is active.

REG 15-18 (Offset 0108h) SPIC_WRITE_ENABLE

Bits	Access	INI	Symbol	Description
7:0	R/W	0x06	WR_EN_CMD	Indicate SPI Flash command value of write enable.

15.3.19 SPIC_READ_STATUS

- Name: Read Status Command Register
- Width: 16 bits
- Initial Value: 0x0505

It is used to implement read status command in auto mode. It can't be programmed when SSIENR is active.

REG 15-19 (Offset 010Ch) SPIC_READ_STATUS

Bits	Access	INI	Symbol	Description
7:0	R/W	0x05	RD_ST_CMD	Indicate SPI Flash command value of read status register. (If the target flash has more than one status registers, this command is used to read the status register 1)

15.3.20 SPIC_CTRLR2

- Name: Control Register 2
- Width: 14 bits
- Initial Value: 0x661

It is used to define SPIC hardware status. Through programming the register, it is flexible to modify the hardware connection with flash and some hardware features. It can't be programmed when SSIENR is active.

REG 15-20 (Offset 0110h) SPIC_CTRLR2

Bits	Access	INI	Symbol	Description
3	R/W	0x0	SEQ_EN	Set to enable Data-Split Program. SPIC keeps transmit data to flash as long as CPU pushes data into SPIC's FIFO. If CPU cannot push data in time but SPIC still sends clocks, unknown value may be programmed or page program may fail. Enable this bit can remain chip select to low before all data are successfully pushed to FIFO even when CPU cannot push data in time. Must disable this bit when page program is done.

15.3.21 SPIC_FBAUDR

- Name: Fast Read Baud Rate Register
- Width: 12 bits
- Initial Value: 0x0001

It is used to implement different baud rate from BAUDR register. It is especially for fast read command. If FBAUDR is setting 0 and implementing fast read command, SPIC would halt in IDLE state. It can't be programmed when SSIENR is active.

REG 15-21 (Offset 0114h) SPIC_FBAUDR

Bits	Access	INI	Symbol	Description
11:0	R/W	0x1	FSCKDV	Indicate the divider of Fast read command when FAST_RD is setting in user mode or using fast read command in auto mode. The frequency of spi_sclk is derived from: Frequency of spi_sclk = Frequency of spic_clk / (2*FSCKDV).

15.3.22 SPIC_ADDR_LENGTH

- Name: Address Bye Length Register
- Width: 3 bits

■ Initial Value: 0x03

It is used in user mode. It determines byte numbers of address phase. The address phase is followed by the command phase. For example, the write status register command should set this register to 1 indicating the address byte is one. It can't be programmed when SSIENR is active.

REG 15-22 (Offset 0118h) SPIC_ADDR_LENGTH

Bits	Access	INI	Symbol	Description
2:0	R/W	0x3	ADDR_PHASE_LENGTH	Indicate byte numbers of address phase (between command phase and write/read phase) in user mode. AUTO_ADDR_LENGTH should be 1, 2, 3, or 0 If it is set to 0, it would set AUTO_ADDR_LENGTH to "4" bytes to support 4-byte address mode.

15.3.23 SPIC_AUTO_LENGTH

- Name: Auto Mode Address Byte Length Register
- Width: 32 bits
- Initial Value: 0x58030000

It decides the delay cycles to receive data in auto/user mode and byte number of address in read/write auto command. It can't be programmed when SSIENR is active.

REG 15-23 (Offset 011C) SPIC_AUTO_LENGTH

Bits	Access	INI	Symbol	Description
31:28	R/W	CS_H_WR_DUM	CS_H_WR_DUM_LEN	Dummy cycle between sending write command to SPI Flash. Using the dummy cycles to avoid the timing violation of CS high time. The value of CS_H_RD_DUM_LEN is a switch to select CS high Read/Write time in a fine grain or in a coarse grain. Fine grain (CS_H_RD_DUM_LEN != 0) CS high write time = CS_H_WR_DUM_LEN * bus clock period. Coarse grain (CS_H_RD_DUM_LEN == 0, value is a multiple of 4) CS high write time = CS_H_WR_DUM_LEN * bus clock period * 4
27:26	R/W	CS_H_RD_DUM	CS_H_RD_DUM_LEN	Dummy cycle between sending read command to SPI Flash. Using the dummy cycles to avoid the timing violation of CS high time. The value of CS_H_RD_DUM_LEN is a switch to select CS high Read/Write time in a fine grain or in a coarse grain. Fine grain (CS_H_RD_DUM_LEN != 0) CS high read time = CS_H_RD_DUM_LEN * bus clock period. Coarse grain (CS_H_RD_DUM_LEN == 0, value is a multiple of 4) CS high write time = CS_H_WR_DUM_LEN * bus clock period * 4
25:18	R/W	0x0	AUTO_DUM_LEN	A dummy cycle is used to check flash status in auto_write operation if the delay time of read data > 1 cycle.
17:16	R/W	0x3	AUTO_ADDR_LENGTH	Indicate byte numbers of address in read/write command in auto mode. AUTO_ADDR_LENGTH should be 1, 2, or 3 bytes. If it is set to 0, it would set AUTO_ADDR_LENGTH to "4" bytes to support 4-byte address mode.

				support 4-byte address mode. Note: If PRM in auto mode (continuous read mode) is enabled by setting VALID_CMDSPIC_11], AUTO_ADDR_LENGTH should be modified. For example, if the flash is under three byte address mode, it should add one more byte to address phase when PRM mode (continuous read mode) is enable. The additional address byte is used to insert a control parameter of continuous read mode before dummy cycles.
11:0	R/W	0x0	RD_DUMMY_LEN GTH	Indicate delay cycle for receiving data. It is regulated by oc_clk (i.e. spic_clk). $RD_DUMMY_LENGTH = \text{Dummy cycle of Flash} * (\text{clock divider in BAUDR or FBAUDR})^2 + \text{pad delay}$. The dummy cycle of flash is defined in flash datasheets. It is not a fixed value depending on which read commands and which flash types we use. The unit of flash's dummy cycle is spi_sclk, the first term of the formula transforms spi_sclk clock base to spic_clk clock base.

15.3.24 SPIC_VALID_CMD

- Name: Control Register 0
- Width: 15 bits
- Initial Value: 0x0200

The register is used in auto mode. It defines all valid commands for auto mode. Except the WR_BLOCKING bit field that can ensure data is popped from FIFO, each valid command corresponds to different flash commands. We define instructions of these flash commands in the registers mentioned before. Set a particular bit of the valid command register will enable that command in auto mode. If none of these bits is set, the flash controller use READ command (0x03h) and Page Program command (0x02h) as default commands for auto mode read / write. We have five read commands and four write commands available. Read commands and write commands are configured separately. Users can enable one or more read / write commands at the same time, but only one read / write command is executed. For example, if RD_DUAL_I and RD_QUAD_O are enabled simultaneously, the flash controller automatically executes the most efficient command and ignores the other one. In this case, RD_QUAD_O is used in auto mode read operation. Note that the flash controller cannot accept next command until current auto mode command is complete.

In addition to those read / write commands, three particular bits provide special functions to access flash. PRM_EN enables continuous read mode for some flash types, it allows consecutive read commands to be executed without sending the same read instruction for the second and the after read operation.

To be compatible with various flash types, please use API functions to access flash. This register can't be programmed when SSIENR is active.

REG 15-24 (Offset 0120) SPIC_VALID_CMD

Bits	Access	INI	Symbol	Description
11	R/W	0x0	PRM_EN	Set to enable SPIC performance read mode (continuous read mode) in auto mode. Several steps should be configured to enter and exit the performance read mode. Not all flash IO modes support this feature. Please access flash with provided API functions to prevent inadvertent side effect.
9	R/W	0x1	WR_BLOCKING	Set to enable blocking state, SPIC does not accept next operation until data pushed into FIFO is entirely popped out of FIFO. If this bit is unset, SPIC can accept next operation after it only checks the write data pushing to FIFO. Always should be set.
8	R/W	0x0	WR_QUAD_II	Enable quad address/data write (1-4-4) command in auto mode. Not support.
7	R/W	0x0	WR_QUAD_I	Enable quad data write (1-1-4) command in auto mode. Not support.
6	R/W	0x0	WR_DUAL_II	Enable dual address/data write (1-2-2) command in auto mode. Not support.
5	R/W	0x0	WR_DUAL_I	Enable dual data write (1-1-2) command in auto mode. Not support.
4	R/W	0x0	RD_QUAD_IO	Enable quad address/data read (1-4-4) command in auto mode.
3	R/W	0x0	RD_QUAD_O	Enable quad data read (1-4-4) command in auto mode.
2	R/W	0x0	RD_DUAL_IO	Enable dual address/data read (1-2-2) command in auto mode.
1	R/W	0x0	RD_DUAL_I	Enable dual data read (1-1-2) command in auto mode.
0	R/W	0x0	FRD_SINGEL	Enable fast read command in auto mode. SPIC would reference FBAUDR register to generate spi_sclk.

15.3.25 SPIC_FLUSH_FIFO

- Name: Control Register 0
- Width: 1 bits
- Initial Value: 0x0

The register is used to flush data and makes FIFO is empty. It's usually used when the behavior of the controller is abnormal or data is crashed in FIFO. It can't be programmed when SSIENR is active.

REG 15-25 (Offset 0128h) SPIC_FLUSH_FIFO

Bits	Access	INI	Symbol	Description
0	W	0x0	FLUSH_FIFO	Clear all data in the FIFO when writing to FLASU_FIFO register.

16 General Purpose Input/Output (GPIO)

16.1 Overview

16.1.1 Application Scenario

GPIO is a programmable General Purpose Programming I/O peripheral. Ameba-Z II's GPIO IP controls the output data and direction of external I/O pads. And Ameba-Z II supports one GPIO IP: PORT A (0~24). 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:

16.1.2 Features

- In/Out mode
 - Output high
 - Output low
 - Toggle output
 - Input read
- All GPIO support interrupt
 - support dual-edge trigger
 - support rising edge trigger
 - support falling edge trigger
 - support edge trigger
 - support level trigger
- HW de-bounce up to 500ms
 - De-bounce input read
- Schmitt trigger control: enable and level select

16.1.3 Architecture

A simplified block diagram of the component is illustrated in Figure 16-1.

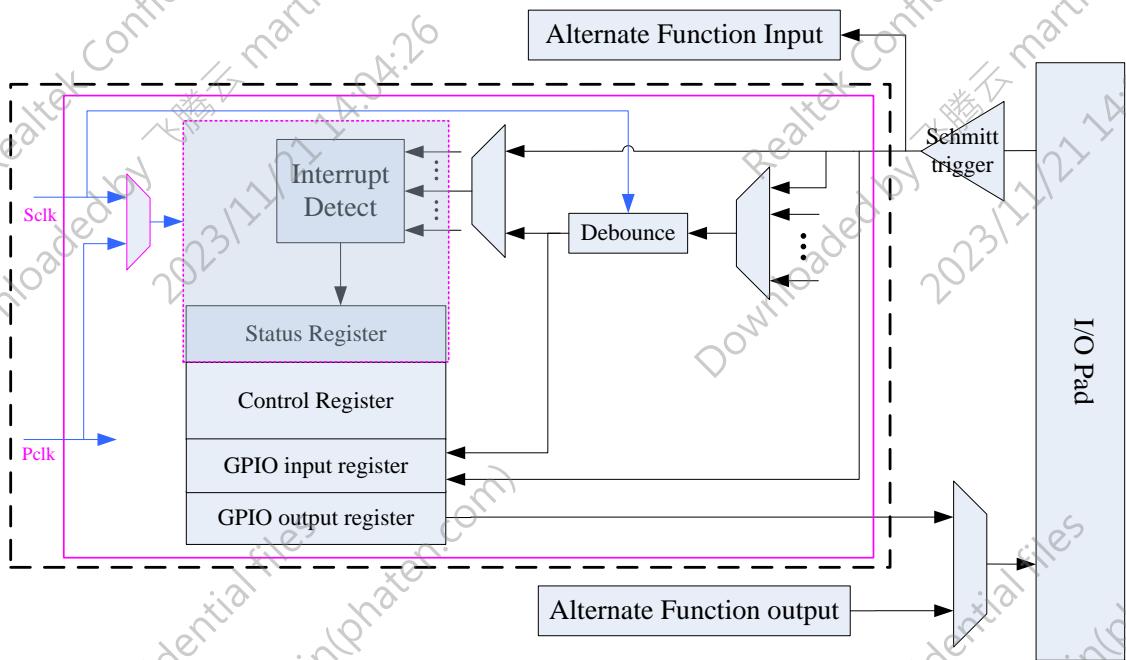


Figure 16-1 GPIO Architecture

16.2 Functional Description

This chapter describes the functional operation of the GPIO.

16.2.1 GPIO Input and Output Control

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.

- **Input Mode:** GPIO input mode can be controlled by the GPIO group A input data mode enable Register (GPIOA_IDM_EN), while the correspond bit was set to “1”. The GPIO as the input data mode and the user can read the input data value by reading GPIO group A data pin status Register (GPIOA_DP_STS).
- **Output Mode:** GPIO output mode can be controlled by the GPIO group A output data mode enable Register (GPIOA_ODM_EN), while the correspond bit was set to “1”. The GPIO as the output data mode and the user can control the below registers to change output level:
 - GPIO group A output data low enable Register (GPIOA_ODL_EN)

- GPIO group A output data high enable Register (GPIOA_ODH_EN)
- GPIO group A output data toggle enable Register (GPIOA_ODT_EN)

16.2.2 Interrupts

GPIO Port A can be programmed to accept external signals as interrupt sources on any of the bits of the signal and support up to 16 interrupt register at the same time. The type of interrupt is programmable with one of the following settings:

- **Active-high and level:** The GPIO active-high and level interrupt will trigger by input "high" level and the following settings shown below:
 - GPIO interrupt rising-edge enable Register (GPIO_IR_EN) was set to "1".
 - GPIO level sensitive interrupt mode enable Register (GPIO_LI_EN) was set to "1".
- **Active-low and level:** The GPIO active-high and level interrupt will trigger by input "low" level and the following settings shown below:
 - GPIO interrupt falling-edge enable Register (GPIO_IF_EN) was set to "1".
 - GPIO level sensitive interrupt mode enable Register (GPIO_LI_EN) was set to "1".
- **Rising edge:** The GPIO rising edge interrupt will trigger by input "low to high" pulse and the following settings shown below:
 - GPIO interrupt rising-edge enable Register (GPIO_IR_EN) was set to "1".
 - GPIO edge sensitive interrupt mode enable Register (GPIO_EI_EN) was set to "1".
- **Falling edge:** The GPIO falling edge interrupt will trigger by input "high to low" pulse and the following settings shown below:
 - GPIO interrupt falling-edge enable Register (GPIO_IF_EN) was set to "1".
 - GPIO edge sensitive interrupt mode enable Register (GPIO_EI_EN) was set to "1".
- **Dual edge:** The GPIO dual edge interrupt will trigger by both input "low to high" or "high to low" pulse and the following settings shown below:
 - GPIO interrupt dual-edge enable Register (GPIO_ID_EN) was set to "1".
 - GPIO edge sensitive interrupt mode enable Register (GPIO_EI_EN) was set to "1".

Note: The correspond interrupts was shown in Figure 16-2.

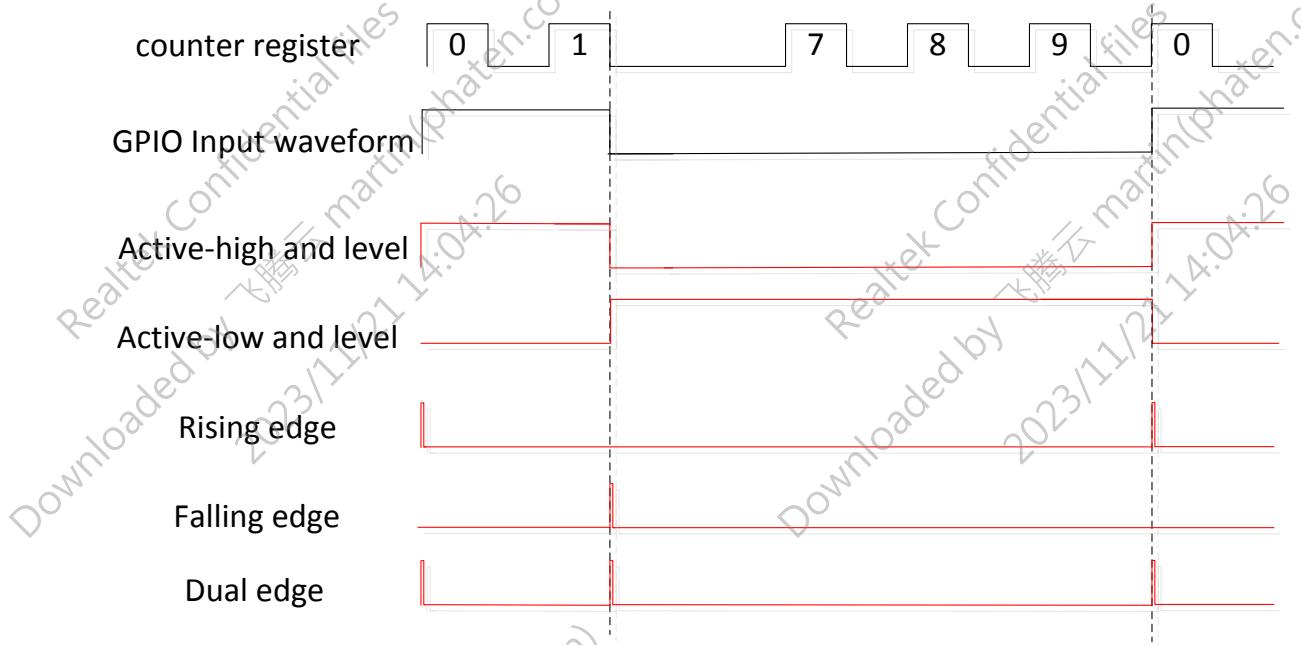


Figure 16-2 The Corresponding GPIO Interrupts

16.2.3 Debounce Operation

If the user has configured GPIO Port A to include the interrupt feature, GPIO can be configured to either include or exclude a debounce capability using the GPIO debouncex selection Register (GPIO_DEBx_SEL). The external signal can be debounced to remove any spurious glitches that are less than one period of the external debouncing clock and support up to 16 debounce register at the same time.

$$\text{Debounce time} = (\text{GPIO_DEBx_CYC}) * T_{\text{Debounce}}$$

$$\text{Where } T_{\text{Debounce}} = 1/32k$$

The Figure 16-3 shows the GPIO debounce operation diagram. The external signal can be debounced to remove any spurious glitches according to the “Debounce time”.

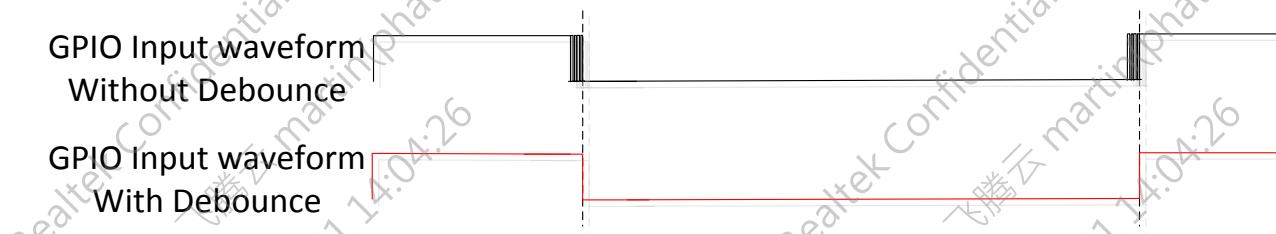


Figure 16-3 GPIO Debounce Operation Diagram

16.3 Control Registers

16.3.1 GPIO_IT_STS

- Name: GPIO interrupt type status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-1 (Offset 0000h) GPIO_IT_STS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R	0	GPIO_IT_STS	0: the specific GPIO port is configured to edge sensitive interrupt mode 1: the specific GPIO port is configured to level sensitive interrupt mode.

16.3.2 GPIO_EI_EN

- Name: GPIO edge sensitive interrupt mode enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-2 (Offset 0004h) GPIO_EI_EN

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_EI_EN	When wrote 0: No operation 1: the controlled GPIO port is configured to edge sensitive interrupt mode and the specific bit of REG_GPIOA_IT_STS is zero.

16.3.3 GPIO_LI_EN

- Name: GPIO level sensitive interrupt mode enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-3 (Offset 0008h) GPIO_LI_EN

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_LI_EN	When wrote 0: No operation 1: the controlled GPIO port is configured to edge sensitive interrupt mode and the specific bit of REG_GPIOA_IT_STS is ONE.

16.3.4 GPIO_IP_STS

- Name: GPIO interrupt polarity status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-4 (Offset 000Ch) GPIO_IP_STS

Bit	Access	INI	Symbol	Description
31:30	R	0	GPIO15_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
29:28	R	0	GPIO14_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
27:26	R	0	GPIO13_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
25:24	R	0	GPIO12_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
23:22	R	0	GPIO11_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
21:20	R	0	GPIO10_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge

				(active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
19:18	R	0	GPIO9_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
17:16	R	0	GPIO8_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
15:14	R	0	GPIO7_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
13:12	R	0	GPIO6_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
11:10	R	0	GPIO5_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
9:8	R	0	GPIO4_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
7:6	R	0	GPIO3_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
5:4	R	0	GPIO2_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
3:2	R	0	GPIO1_IP_STS	00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge

1:0	R	0	GPIO0_IP_STS	11: rsvd 00: the specific GPIO port is configured to rising-edge (active-high sensitive) 01: the specific GPIO port is configured to falling-edge (active-low sensitive) 10: the specific GPIO port is configured to dual-edge 11: rsvd
-----	---	---	--------------	---

16.3.5 GPIO_IR_EN

- Name: GPIO interrupt rising-edge enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-5 (Offset 0014h) GPIO_IR_EN

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_IR_EN	When wrote 0: No operation; 1: the controlled GPIO port is configured to rising-edge interrupt mode and the specific bits of REG_GPIOA_IP_STS is "00".

16.3.6 GPIO_IF_EN

- Name: GPIO interrupt falling-edge enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-6 (Offset 0018h) GPIO_IF_EN

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_IF_EN	When wrote 0: No operation; 1: the controlled GPIO port is configured to falling-edge interrupt mode and the specific bits of REG_GPIOA_IP_STS is "01".

16.3.7 GPIO_ID_EN

- Name: GPIO interrupt dual-edge enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-7 (Offset 001Ch) GPIO_ID_EN

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_ID_EN	When wrote 0: No operation; 1: the controlled GPIO port is configured to dual-edge interrupt mode and the specific bits of REG_GPIOA_IP_STS is "10".

16.3.8 GPIO_IM_STS

- Name: GPIO interrupt mask status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-8 (Offset 0020h) GPIO_IM_STS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R	0	GPIO_IM_STS	0: Disable interrupt 1: Enable interrupt

16.3.9 GPIO_IM_EN

- Name: GPIO enable interrupt mask Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-9 (Offset 0024h) GPIO_IM_EN

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_IM_EN	When wrote 0: No operation 1: the specific GPIO INT is masked and REG_GPIO_IM_STS is ONE.

16.3.10 GPIO_IM_DIS

- Name: GPIO disable interrupt mask Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-10 (Offset 0028h) GPIO_IM_DIS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_IM_DIS	When wrote 0: No operation 1: the controlled GPIO INT is unmasked and the specific bit of REG_GPIO_IM_STS is ZERO.

16.3.11 GPIO_RAW_INT_STS

- Name: RAW interrupt status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-11 (Offset 002Ch) GPIO_RAW_INT_STS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R	0	GPIO_RAW_INT_STS	0: the specific GPIO port has no interrupt request 1: the specific GPIO port has interrupt request (permasking)

16.3.12 GPIO_INT_STS

- Name: GPIO interrupt status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-12 (Offset 0030h) GPIO_INT_STS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R	0	GPIO_INT_STS	0: the specific GPIO port has no interrupt request 1: the specific GPIO port has interrupt request

16.3.13 GPIO_INT_CLR

- Name: **GPIO interrupt status clear Register**
- Width: 32bit
- Initial Value: 0x0000

REG 16-13 (Offset 0034h) GPIO_INT_CLR

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_INT_CLR	When wrote 0: No operation; 1: Clear edge type interrupt request of the controlled GPIO port and the specific bits of REG_GPIOA_INT_STS is ZERO.

16.3.14 GPIO_INT_EN_STS

- Name: GPIO interrupt enable status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-14 (Offset 0038h) GPIO_INT_EN_STS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R	0	GPIO_INT_EN_STS	0: the specific GPIO INT is Disable 1: the specific GPIO INT is Enable

16.3.15 GPIO_INT_EN

- Name: GPIO enable interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-15 (Offset 003Ch) GPIO_INT_EN

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_INT_EN	When wrote 0: No operation 1: the specific GPIO INT is enabled and the specific bit of REG_GPIO_INT_EN_STS is ONE.

16.3.16 GPIO_INT_DIS

- Name: GPIO disable interrupt Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-16 (Offset 0040h) GPIO_INT_DIS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_INT_DIS	When wrote 0: No operation 1: the controlled GPIO INT is disabled and the specific bit of REG_GPIO_INT_EN_STS is ZERO.

16.3.17 GPIO_INT0_SEL

Name: GPIO INTO selection Register

Width: 32bit

- Initial Value: 0x0000

REG 16-17 (Offset 0050h) GPIO_INT0_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INT0_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INT0_DEB_SEL	Debounce Output signal selection for GPIO INTO
7	R	0	RSVD	
6:5	R/W	0	GPIO_INT0_GP_SEL	GPIO group selection for GPIO INTO
4:0	R/W	0	GPIO_INT0_MER_SEL	Member selection of specified group for GPIO INTO

16.3.18 GPIO_INT1_SEL

- Name: GPIO INTO selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-18 (Offset 0054h) GPIO_INT1_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INT1_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INT1_DEB_SEL	Debounce Output signal selection for GPIO INT1
7	R	0	RSVD	
6:5	R/W	0	GPIO_INT1_GP_SEL	GPIO group selection for GPIO INT1
4:0	R/W	0	GPIO_INT1_MER_SEL	Member selection of specified group for GPIO INT1

16.3.19 GPIO_INT2_SEL

- Name: GPIO INT2 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-19 (Offset 0058h) GPIO_INT2_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INT2_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INT2_DEB_SEL	Debounce Output signal selection for GPIO INT2
7	R	0	RSVD	
6:5	R/W	0	GPIO_INT2_GP_SEL	GPIO group selection for GPIO INT2
4:0	R/W	0	GPIO_INT2_MER_SEL	Member selection of specified group for GPIO INT2

16.3.20 GPIO_INT3_SEL

- Name: GPIO INT3 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-20 (Offset 005Ch) GPIO_INT3_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INT3_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INT3_DEB_SEL	Debounce Output signal selection for GPIO INT3
7	R	0	RSVD	
6:5	R/W	0	GPIO_INT3_GP_SEL	GPIO group selection for GPIO INT3
4:0	R/W	0	GPIO_INT3_MER_SEL	Member selection of specified group for GPIO INT3

16.3.21 GPIO_INT4_SEL

- Name: GPIO INT4 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-21 (Offset 0060) GPIO_INT4_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INT4_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INT4_DEB_SEL	Debounce Output signal selection for GPIO INT4
7	R	0	RSVD	
6:5	R/W	0	GPIO_INT4_GP_SEL	GPIO group selection for GPIO INT4
4:0	R/W	0	GPIO_INT4_MER_SEL	Member selection of specified group for GPIO INT4

16.3.22 GPIO_INT5_SEL

- Name: GPIO INT5 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-22 (Offset 0064) GPIO_INT5_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INT5_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INT5_DEB_SEL	Debounce Output signal selection for GPIO INT5
7	R	0	RSVD	
6:5	R/W	0	GPIO_INT5_GP_SEL	GPIO group selection for GPIO INT5
4:0	R/W	0	GPIO_INT5_MER_SEL	Member selection of specified group for GPIO INT5

16.3.23 GPIO_INT6_SEL

- Name: GPIO INT6 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-23 (Offset 0068) GPIO_INT6_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INT6_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INT6_DEB_SEL	Debounce Output signal selection for GPIO INT6
7	R	0	RSVD	

6:5	R/W	0	GPIO_INT6_GP_SEL	GPIO group selection for GPIO INT6
4:0	R/W	0	GPIO_INT6_MER_SEL	Member selection of specified group for GPIO INT6

16.3.24 GPIO_INT7_SEL

- Name: GPIO INT7 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-24 (Offset 006C) GPIO_INT4_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INT7_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INT7_DEB_SEL	Debounce Output signal selection for GPIO INT7
7	R	0	RSVD	
6:5	R/W	0	GPIO_INT7_GP_SEL	GPIO group selection for GPIO INT7
4:0	R/W	0	GPIO_INT7_MER_SEL	Member selection of specified group for GPIO INT7

16.3.25 GPIO_INT8_SEL

- Name: GPIO INT8 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-25 (Offset 0070) GPIO_INT4_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INT8_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INT8_DEB_SEL	Debounce Output signal selection for GPIO INT8
7	R	0	RSVD	
6:5	R/W	0	GPIO_INT8_GP_SEL	GPIO group selection for GPIO INT8
4:0	R/W	0	GPIO_INT8_MER_SEL	Member selection of specified group for GPIO INT8

16.3.26 GPIO_INT9_SEL

- Name: GPIO INT9 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-26 (Offset 0074) GPIO_INT9_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INT9_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INT9_DEB_SEL	Debounce Output signal selection for GPIO INT9
7	R	0	RSVD	
6:5	R/W	0	GPIO_INT9_GP_SEL	GPIO group selection for GPIO INT9
4:0	R/W	0	GPIO_INT9_MER_SEL	Member selection of specified group for GPIO INT9

16.3.27 GPIO_INTA_SEL

- Name: GPIO INTA selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-27 (Offset 0078) GPIO_INTA_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INTA_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INTA_DEB_SEL	Debounce Output signal selection for GPIO INTA
7	R	0	RSVD	
6:5	R/W	0	GPIO_INTA_GP_SEL	GPIO group selection for GPIO INTA
4:0	R/W	0	GPIO_INTA_MER_SEL	Member selection of specified group for GPIO INTA

16.3.28 GPIO_INTB_SEL

- Name: GPIO INTB selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-28 (Offset 007C) GPIO_INTB_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INTB_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INTB_DEB_SEL	Debounce Output signal selection for GPIO INTB
7	R	0	RSVD	
6:5	R/W	0	GPIO_INTB_GP_SEL	GPIO group selection for GPIO INTB
4:0	R/W	0	GPIO_INTB_MER_SEL	Member selection of specified group for GPIO INTB

16.3.29 GPIO_INTC_SEL

- Name: GPIO INTC selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-29 (Offset 0080) GPIO_INTC_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INTC_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INTC_DEB_SEL	Debounce Output signal selection for GPIO INTC
7	R	0	RSVD	
6:5	R/W	0	GPIO_INTC_GP_SEL	GPIO group selection for GPIO INTC
4:0	R/W	0	GPIO_INTC_MER_SEL	Member selection of specified group for GPIO INTC

16.3.30 GPIO_INTD_SEL

- Name: GPIO INTD selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-30 (Offset 0084) GPIO_INTD_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INTD_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INTD_DEB_SEL	Debounce Output signal selection for GPIO INTD
7	R	0	RSVD	

6:5	R/W	0	GPIO_INTD_GP_SEL	GPIO group selection for GPIO INTD
4:0	R/W	0	GPIO_INTD_MER_SEL	Member selection of specified group for GPIO INTD

16.3.31 **GPIO_INTE_SEL**

- Name: GPIO INTE selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-31 (Offset 0088) GPIO_INTE_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INTE_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INTE_DEB_SEL	Debounce Output signal selection for GPIO INTE
7	R	0	RSVD	
6:5	R/W	0	GPIO_INTE_GP_SEL	GPIO group selection for GPIO INTE
4:0	R/W	0	GPIO_INTE_MER_SEL	Member selection of specified group for GPIO INTE

16.3.32 **GPIO_INTF_SEL**

- Name: GPIO INTF selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-32 (Offset 008C) GPIO_INTF_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	GPIO_INTF_SUR_SEL	0: Using pin input signal as source 1: Using debounce output signal as source
14:12	R	0	RSVD	
11:8	R/W	0	GPIO_INTF_DEB_SEL	Debounce Output signal selection for GPIO INTF
7	R	0	RSVD	
6:5	R/W	0	GPIO_INTF_GP_SEL	GPIO group selection for GPIO INTF
4:0	R/W	0	GPIO_INTF_MER_SEL	Member selection of specified group for GPIO INTF

16.3.33 **GPIO_DEB_STS**

- Name: GPIO group debounce status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-33 (Offset 00F0) GPIO_DEB_STS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R	0	GPIO_DEB_STS	0: the specific GPIO debounce port is DISABLE debounce 1: the specific GPIO debounce port is ENABLE debounce

16.3.34 **GPIO_DEB_EN**

- Name: GPIO debounce port enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-34 (Offset 00F4) GPIO_DEB_EN

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_DEB_EN	When wrote 0: No operation 1: the controlled GPIO debounce port is enable debounce and the specific bit of REG_GPIO_DEB_STS is ONE.

16.3.35 **GPIO_DEB_DIS**

- Name: GPIO debounce port disable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-35 (Offset 00F8) GPIO_DEB_DIS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	WO	0	GPIO_DEB_DIS	When wrote 0: No operation 1: the controlled GPIO debounce port is disable debounce and the specific bit of REG_GPIO_DEB_STS is ZERO.

16.3.36 DEB_DP_STS

- Name: GPIO debounce data pin status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-36 (Offset 00FC) DEB_DP_STS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:0	R/W	0	GPIO_DEB_DP_STS	Reading this location reads the values on the signal of specified pin after debounce.

16.3.37 GPIO_DEBO_SEL

- Name: GPIO debounce0 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-37 (Offset 0100) GPIO_DEBO_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEBO_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEBO_GP_SEL	GPIO group selection for debounce0
4:0	R/W	0	GPIO_DEBO_MER_SEL	Member selection of specified group for debounce0

16.3.38 GPIO_DEB1_SEL

- Name: GPIO debounce1 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-38 (Offset 0104) GPIO_DEB1_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEB1_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEB1_GP_SEL	GPIO group selection for debounce1

4:0	R/W	0	GPIO_DEB1_MER_SEL	Member selection of specified group for debounce1
-----	-----	---	-------------------	---

16.3.39 GPIO_DEB2_SEL

- Name: GPIO debounce2 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-39 (Offset 0108) GPIO_DEB2_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEB2_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEB2_GP_SEL	GPIO group selection for debounce2
4:0	R/W	0	GPIO_DEB2_MER_SEL	Member selection of specified group for debounce2

16.3.40 GPIO_DEB3_SEL

- Name: GPIO debounce3 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-40 (Offset 010C) GPIO_DEB3_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEB3_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEB3_GP_SEL	GPIO group selection for debounce3
4:0	R/W	0	GPIO_DEB3_MER_SEL	Member selection of specified group for debounce3

16.3.41 GPIO_DEB4_SEL

- Name: GPIO debounce4 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-41 (Offset 0110) GPIO_DEB4_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEB4_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEB4_GP_SEL	GPIO group selection for debounce4
4:0	R/W	0	GPIO_DEB4_MER_SEL	Member selection of specified group for debounce4

16.3.42 GPIO_DEB5_SEL

- Name: GPIO debounce5 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-42 (Offset 0114) GPIO_DEB5_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEB5_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEB5_GP_SEL	GPIO group selection for debounce5
4:0	R/W	0	GPIO_DEB5_MER_SEL	Member selection of specified group for debounce5

16.3.43 GPIO_DEB6_SEL

- Name: GPIO debounce6 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-43 (Offset 0118) GPIO_DEB6_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEB6_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEB6_GP_SEL	GPIO group selection for debounce6
4:0	R/W	0	GPIO_DEB6_MER_SEL	Member selection of specified group for debounce6

16.3.44 GPIO_DEB7_SEL

- Name: GPIO debounce7 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-44 (Offset 011C) GPIO_DEB7_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEB7_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEB7_GP_SEL	GPIO group selection for debounce7
4:0	R/W	0	GPIO_DEB7_MER_SEL	Member selection of specified group for debounce7

16.3.45 GPIO_DEB8_SEL

- Name: GPIO debounce8 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-45 (Offset 0120) GPIO_DEB8_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEB8_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEB8_GP_SEL	GPIO group selection for debounce8
4:0	R/W	0	GPIO_DEB8_MER_SEL	Member selection of specified group for debounce8

16.3.46 GPIO_DEB9_SEL

- Name: GPIO debounce9 selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-46 (Offset 0124) GPIO_DEB9_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEB9_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEB9_GP_SEL	GPIO group selection for debounce9
4:0	R/W	0	GPIO_DEB9_MER_SEL	Member selection of specified group for debounce9

16.3.47 GPIO_DEBA_SEL

- Name: GPIO debounceA selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-47 (Offset 0128) GPIO_DEBA_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEBA_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEBA_GP_SEL	GPIO group selection for debounceA
4:0	R/W	0	GPIO_DEBA_MER_SEL	Member selection of specified group for debounceA

16.3.48 GPIO_DEBB_SEL

- Name: GPIO debounceB selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-48 (Offset 012C) GPIO_DEBB_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEBB_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEBB_GP_SEL	GPIO group selection for debounceB
4:0	R/W	0	GPIO_DEBB_MER_SEL	Member selection of specified group for debounceB

16.3.49 GPIO_DEBC_SEL

- Name: GPIO debounceC selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-49 (Offset 0130) GPIO_DEBC_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEBC_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEBC_GP_SEL	GPIO group selection for debounceC
4:0	R/W	0	GPIO_DEBC_MER_SEL	Member selection of specified group for debounceC

16.3.50 GPIO_DEBD_SEL

- Name: GPIO debounceD selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-50 (Offset 0134) GPIO_DEBD_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEBD_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEBD_GP_SEL	GPIO group selection for debounceD
4:0	R/W	0	GPIO_DEBD_MER_SEL	Member selection of specified group for debounceD

16.3.51 GPIO_DEBE_SEL

- Name: GPIO debounceE selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-51 (Offset 0138) GPIO_DEBE_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEBE_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEBE_GP_SEL	GPIO group selection for debounceE
4:0	R/W	0	GPIO_DEBE_MER_SEL	Member selection of specified group for debounceE

16.3.52 GPIO_DEBF_SEL

- Name: GPIO debounceF selection Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-52 (Offset 013C) GPIO_DEBF_SEL

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
21:8	R/W	0	GPIO_DEBF_CYC	The GPIO signal will be filtered by the number of debounce cycles of specified in this register.
7	R	0	RSVD	
6:5	R/W	0	GPIO_DEBF_GP_SEL	GPIO group selection for debounceF
4:0	R/W	0	GPIO_DEBF_MER_SEL	Member selection of specified group for debounceF

16.3.53 GPIOA_DMD_STS

- Name: GPIO group A data mode direction status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-53 (Offset 0200) GPIOA_DMD_STS

Bit	Access	INI	Symbol	Description
31:0	R	0	GPIOA_DMD_STS	0: the specific GPIO port is configured to INPUT data mode 1: the specific GPIO port is configured to OUTPUT data mode , if the specific bit of REG_GPIOA_MODE_STS is 0.

16.3.54 GPIOA_IDM_EN

- Name: GPIO group A input data mode enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-54 (Offset 0204) GPIOA_IDM_EN

Bit	Access	INI	Symbol	Description
31:0	WO	0	GPIOA_IDM_EN	When wrote 0: the controlled GPIO port's configuration is unchanged; 1: the controlled GPIO port is configured to input data mode and the specific bit of REG_GPIOA_DMD_STS is zero.

16.3.55 GPIOA_ODM_EN

- Name: GPIO group A output data mode enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-55 (Offset 0208) GPIOA_ODM_EN

Bit	Access	INI	Symbol	Description
31:0	WO	0	GPIOA_ODM_EN	When wrote 0: the controlled GPIO port's configuration is unchanged; 1: the controlled GPIO port is configured to output data mode and the specific bit of REG_GPIOA_DMD_STS is one.

16.3.56 GPIOA_OD_STS

- Name: GPIO group A output data status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-56 (Offset 020C) GPIOA_OD_STS

Bit	Access	INI	Symbol	Description
31:0	R	0	GPIOA_OD_STS	0: the specific GPIO port is configured to output low 1: the specific GPIO port is configured to output high , if the specific GPIO port is output data mode.

16.3.57 GPIOA_ODL_EN

- Name: GPIO group A output data low enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-57 (Offset 0210) GPIOA_ODL_EN

Bit	Access	INI	Symbol	Description
31:0	WO	0	GPIOA_ODL_EN	When wrote 0: No operation 1: the controlled GPIO port is configured to output low and the specific bit of REG_GPIOA_OD_STS is zero.

16.3.58 GPIOA_ODH_EN

- Name: GPIO group A output data high enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-58 (Offset 0214) GPIOA_ODH_EN

Bit	Access	INI	Symbol	Description
31:0	WO	0	GPIOA_ODH_EN	When wrote 0: No operation 1: the controlled GPIO port is configured to output high and the specific bit of REG_GPIOA_OD_STS is ONE.

16.3.59 GPIOA_ODT_EN

- Name: GPIO group A output data toggle enable Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-59 (Offset 0218) GPIOA_ODT_EN

Bit	Access	INI	Symbol	Description
31:0	WO	0	GPIOA_ODT	When wrote 0: No operation 1: Toggle output of the controlled GPIO port and the specific bit of REG_GPIOA_OD_STS is also toggled.

16.3.60 GPIOA_DP_STS

- Name: GPIO group A data pin status Register
- Width: 32bit
- Initial Value: 0x0000

REG 16-60 (Offset 021C) GPIOA_DP_STS

Bit	Access	INI	Symbol	Description
31:0	R	0	GPIOA_DP_STS	When the specific bit is configured as input mode, then reading this location reads the values on the signal of pin. When the direction of specific bit is configured as output, then reading this location reads the output data register for group A.

17 Security Code Engine (SCE)

17.1 Overview

17.1.1 Application Scenario

In the current time, more and more people emphasize the protection of the code. Then, this chip provides the external memory flash. If there isn't any scheme to protect the code, everyone can capture the code easily. The customers will feel afraid that their competitors steal their code if they adopt this chip to implement their products. For protecting the code of the customers, this chip provides the secure solution which is called "Security Code Engine" (SCE). When the firmware was loaded into the external memory, SCE will encrypt the code with AES128. After initialing process, when CPU wants to fetch the instructions from the external memory, SCE can decrypt the instruction on the fly. By the way, SCE just spends the low latency to decrypt.

17.1.2 Features

- Schmitt trigger control: enable and level select
- SCE provides the below features
 - Just contribute the low latency for decryption
 - Adopt AES128 encryption/decryption
 - Provide 2 set of the key pair
 - One of the key pair: AES key
 - The other of the key pair: IV key
 - The 2 set of key pairs can satisfy two kinds of users.
- Provide 8 secure sections for the secure code; the codes, which are not put into these sections, are non-protected code. Every section can be configured into the difference range individually. The constraint of range is based on the page size configured.
- Remap address for software

17.1.3 Architecture

The simple architecture is shown the below

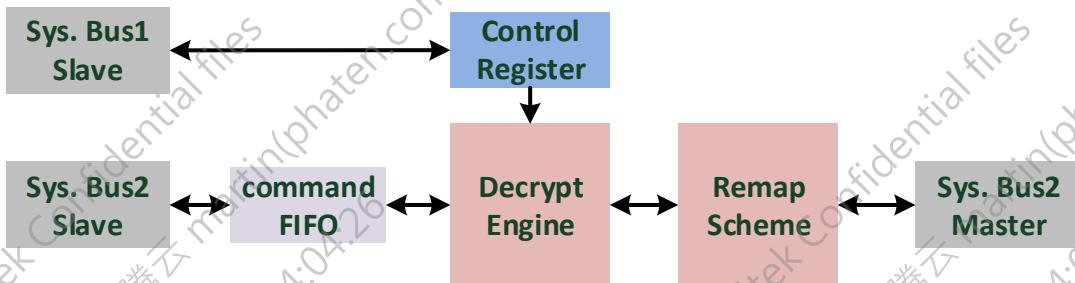


Figure 17-1 The Architecture of “Secure Code Engine”

17.2 Functional Description

This chapter describes the functional operation of SCE.

17.2.1 Decrypt Control Flow

In this section, the preparation part and decrypting part will be introduced below.

Preparation:

Before the SCE can work, the image installed into flash should be encrypted by tool. And the image looks like the below.

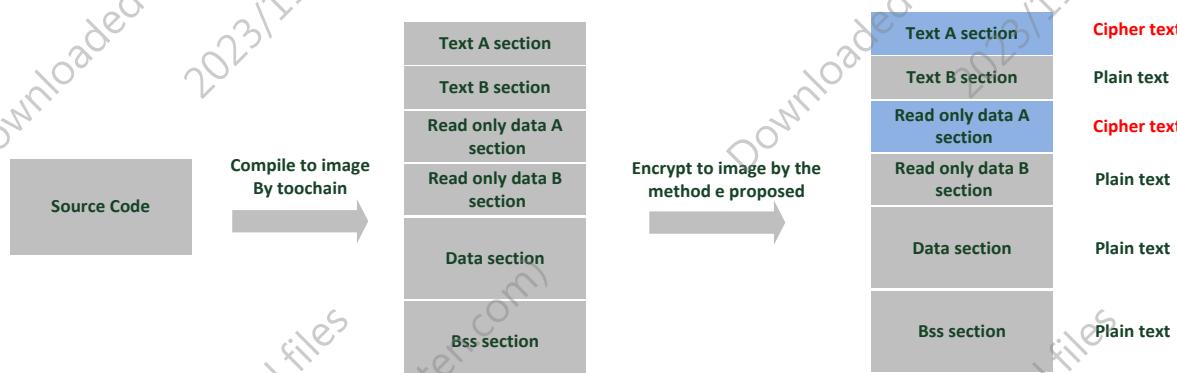


Figure 17-2 The Flow of Generating Secure Image

According to the setting of the tool, it will generate the cipher text and plain text. This scheme just focuses on instruction and read-only data.

On Fly Decrypting Flow:

SCE adopts the block, page and section to present the address. Every block can be 32/64/128 bytes and this block will be decrypted in the one key. Every page can be

16K/32K/64K and this setting can simplify the hardware. Several pages can be a section and every section can be set by different key.

- Secure Block: Every secure section is separated by SCE_BLOCK_SIZE. And SCE will adopt the same key to encrypt or decrypt every secure block. The same key is calculated AES key, IV key and the base address of the secure block.
- Secure Page: SCE separates the whole memory and assign the number. Based on the page number, software can configure the start page and the end page to make the secure section.
- Secure Section: the range of secure code for the users.

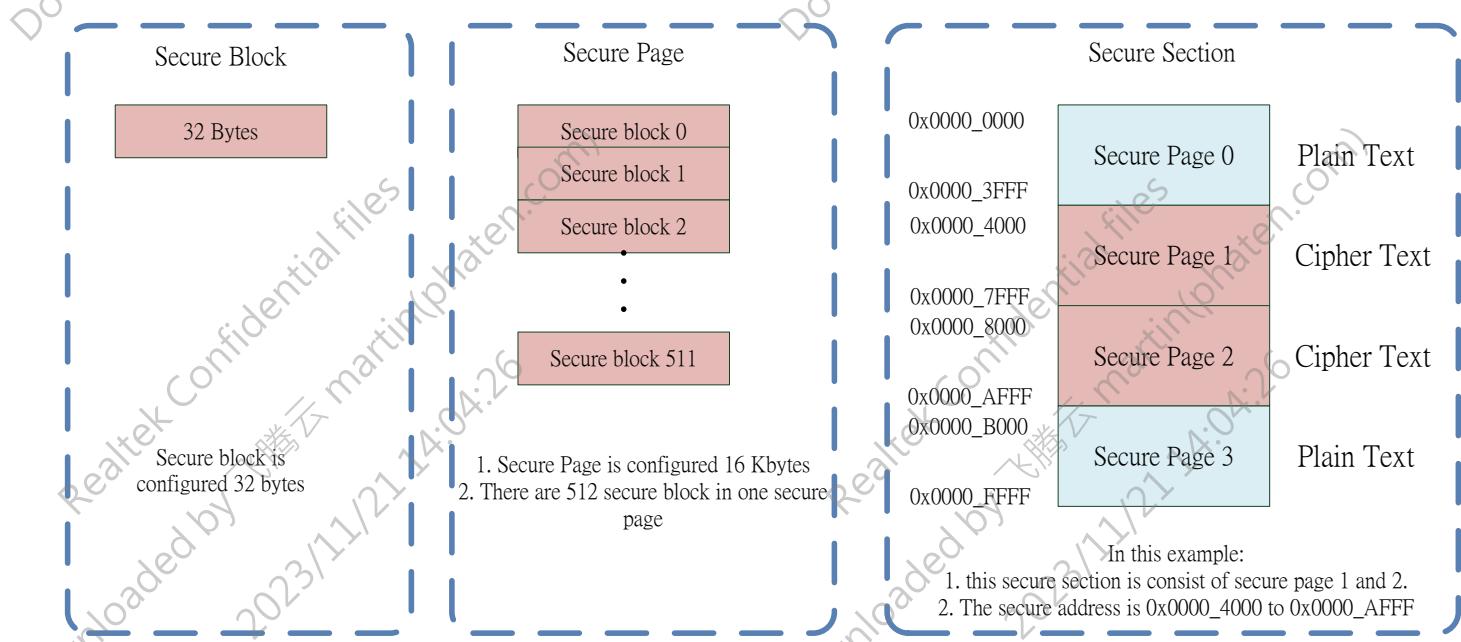


Figure 17-3 The Expression of The Proper Noun for “Secure Code Engine”

When CPU fetch the instructions, how the SCE works is shown the below.

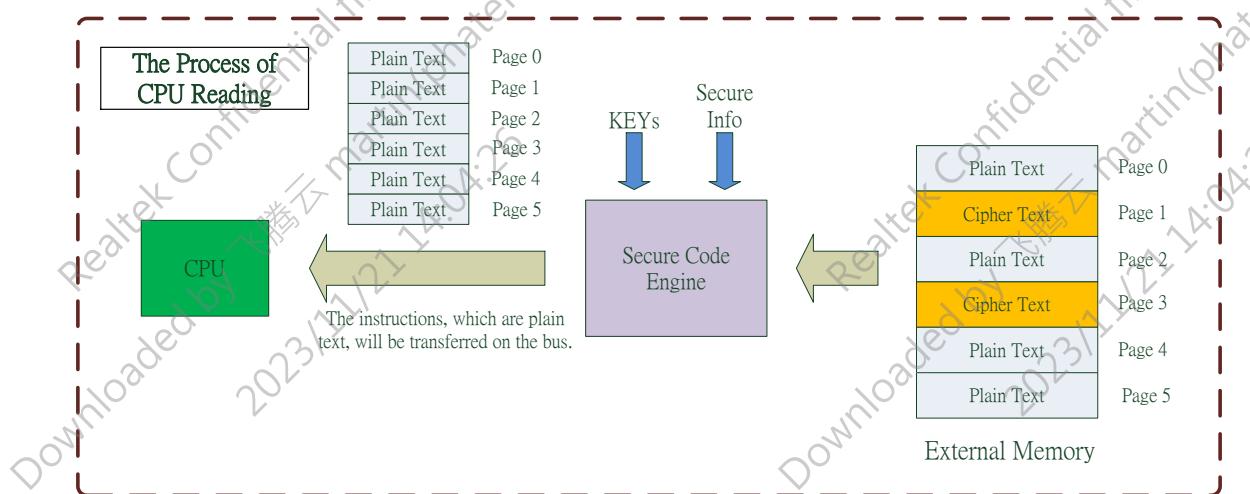


Figure 17-4 The Process of CPU Fetching Instructions by Secure Code Engine

17.2.2 Remap Function

In the normal scenario, users will choose that the instruction can be executed in the flash. Actually, this is XIP mode. Even if users want to update their image, they just want to maintain one address range. Base on this, SCE provides the remap function to solve this. Besides the secure setting, users just need set remap page number in the section. The below figure shows the example that SCE remap the virtual address to physical address.

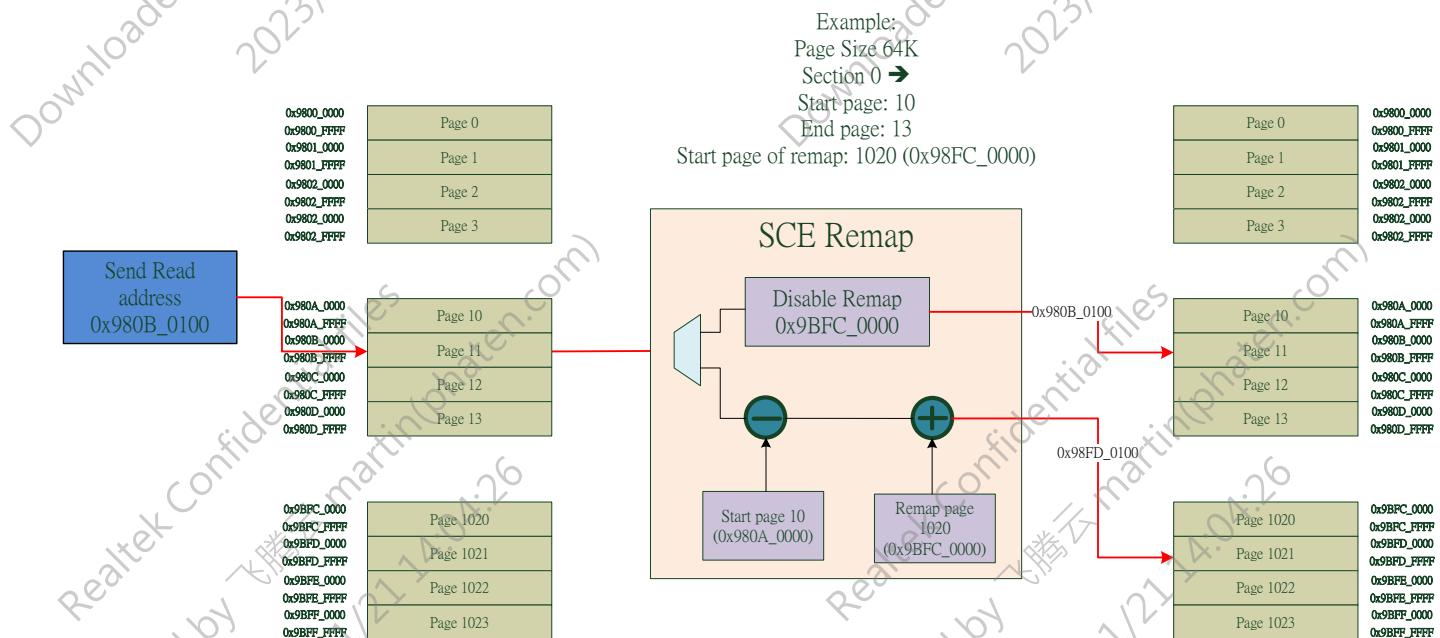


Figure 17-5 The Method of Remap for Secure Code Engine

18 Crypto Engine

18.1 Overview

18.1.1 Application Scenario

The Hardware Crypto Engine on the Ameba-Z II Platform accelerates applications that need cryptographic functions, such as authentication, encryption and decryption. Hardware Crypto engine executing these functions can not only reduce software overhead but also save CPU and Memory resources. And the processing of it is more secure and faster than software.

18.1.2 Features

Crypto engine provides basic cryptographic features that are shown below:

- Crypto Engine
 - Authentication algorithms
 - ◆ General cryptographic hash function
 - ✓ MD5
 - ✓ SHA1
 - ✓ SHA2-224
 - ✓ SHA2-256
 - ◆ HMAC(Hash-based message authentication code)
 - ✓ HMAC_MD5
 - ✓ HMAC_SHA1
 - ✓ HMAC_SHA2-224
 - ✓ HMAC_SHA2-256
 - Cipher(Encryption/Decryption) algorithms
 - ◆ AES-128/192/256
 - ✓ ECB (Electronic Codebook) mode
 - ✓ CBC (Cipher Block Chaining) mode
 - ✓ CTR (Counter) mode
 - ✓ CFB (Cipher Feedback) mode
 - ✓ OFB (Output Feedback) mode
 - ✓ GCTR (Galois CTR) mode
 - ✓ GMAC (Galois MAC) mode
 - ✓ GHASH (Galois HASH) mode

- ✓ GCM (Galois/Counter Mode) mode
- SSH/ESP/SSL-TLS mode
 - ◆ SSH encryption/decryption
 - ◆ ESP encryption/decryption
 - ◆ SSL-TLS encryption

<NOTE>

- Support Once Process Way or Sequential Hash Mechanism in authentication algorithms. If the message length doesn't exceed 1048560 bytes ($16 * (2^{16} - 1)$), we could use Once Process Way to verify the hash function. If the message length exceeds 1048560 bytes ($16 * (2^{16} - 1)$) or we want to divide this message into many particular size blocks to process them, we could use Sequential Hash Mechanism to verify the hash function.
- Support encryption and decryption in Cipher algorithms. Need to care that AES only handles 16bytes aligned message length. If the message length is unaligned, users must auto-padding the rest of bytes.

18.1.3 Architecture

There are 6 main parts inside the Crypto engine architecture. Control registers, CRC engine, Source/Destination FIFO descriptors, Packet-base arbiter, DMA engine, Crypto engine. The programmer can configure or get more detail information of Crypto engine via directly setting or reading control registers. For example: setting basic cryptographic function parameters, setting/reading Crypto interrupt status, setting/reading Crypto Error interrupt, setting DMA/Crypto engine. The more setting detail can reference Chapter 18.3 Control Register. A simplified block diagram of the component is illustrated in Figure 18-1.

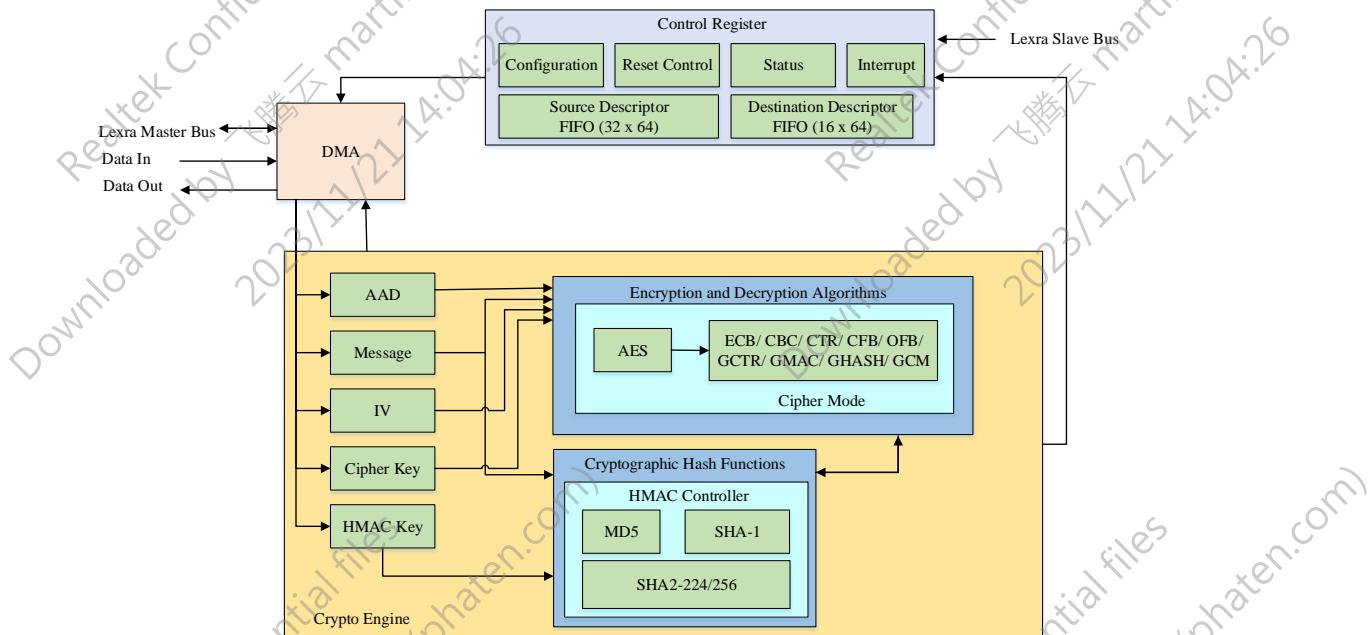


Figure 18-1 Crypto Engine Block Diagram

18.1.3.1 Functional Block Diagram with Clock Domain

Crypto engine implements many kinds of cryptographic algorithm in Crypto engine. For programmer, the way of setting basic cryptographic function parameters is writing data into Source/Destination descriptor first/second word register. The difference between Source descriptor and Destination descriptor is the data buffer that the Crypto engine wants to read/write. Programmers can reference 18.2.3 Descriptor Data Structures to know setting details. Because there are many parameters that need to set, the programmer can disassemble a Source/Destination packet command into some hardware FIFO in Source/Destination descriptors.

- Source descriptor first/second word register: It is used to set cryptographic function input parameters (HMAC Key PAD/KEY/IV/Plaintext buffer).
- Destination descriptor first/second word register: It is used to set cryptographic function output parameters (Digest/Cipher result buffer).

DMA engine gets buffer address from the Source/Destination Descriptor FIFO node, then it access the address. It moves data into Crypto engine before Crypto engine starts to calculate, and when Crypto engine finishes the calculation, it will help move the result data to the result buffer. The more setting details of it, programmers can reference 18.2.3.

18.2 Functional Description

18.2.1 Overview

The major control of Crypto engine is via directly configuring crypto Command/Status registers. However, the configuration of crypto algorithms (Hash/Cipher/Mix mode) is following the descriptor setting mechanism and descriptor data structures, and then setting information to descriptors registers.

18.2.2 Descriptor Setting Mechanism

18.2.2.1 Descriptor FIFO Architecture

18.2.2.1.1 Source Descriptor FIFO

Source Descriptor FIFO gets 32 FIFO nodes. Each FIFO node can point to 2 words (32bits) that are First word and Second word. Use First word to set header information and Second word to point to the address of data buffer. Source Descriptor FIFO uses 2 hardware pointers which are read pointer and write pointer to maintain this FIFO.

Users can set Crypto engine to execute a cryptographic feature via setting a Source Packet Command. A Source Packet Command starts from a descriptor which *FS*=1 and *CL*=3, then ends this packet in a descriptor which *LS*=1. It means that user need to set Command setting first then go on setting others.

When set a Source Packet Command, it just needs to set related information and the setting sequence must follow the figure. For example, if user wants to set Crypto engine to execute Hash algorithms, it doesn't need to set *Key Array* and *IV Array*.

- **Key Array:** *KL* uses 4bytes as a unit, keep previous key when user doesn't set this.
- **IV Array:** *IL* uses 4bytes as a unit, keep previous *IV* when user doesn't set this.
- **PAD Array:** *PL* uses 4bytes as a unit, keep previous *PAD* when user doesn't set this.
- **Data Array:** *A2EO/EPL/ENL/APL* uses 1byte as a unit.

These concepts are illustrated in below:

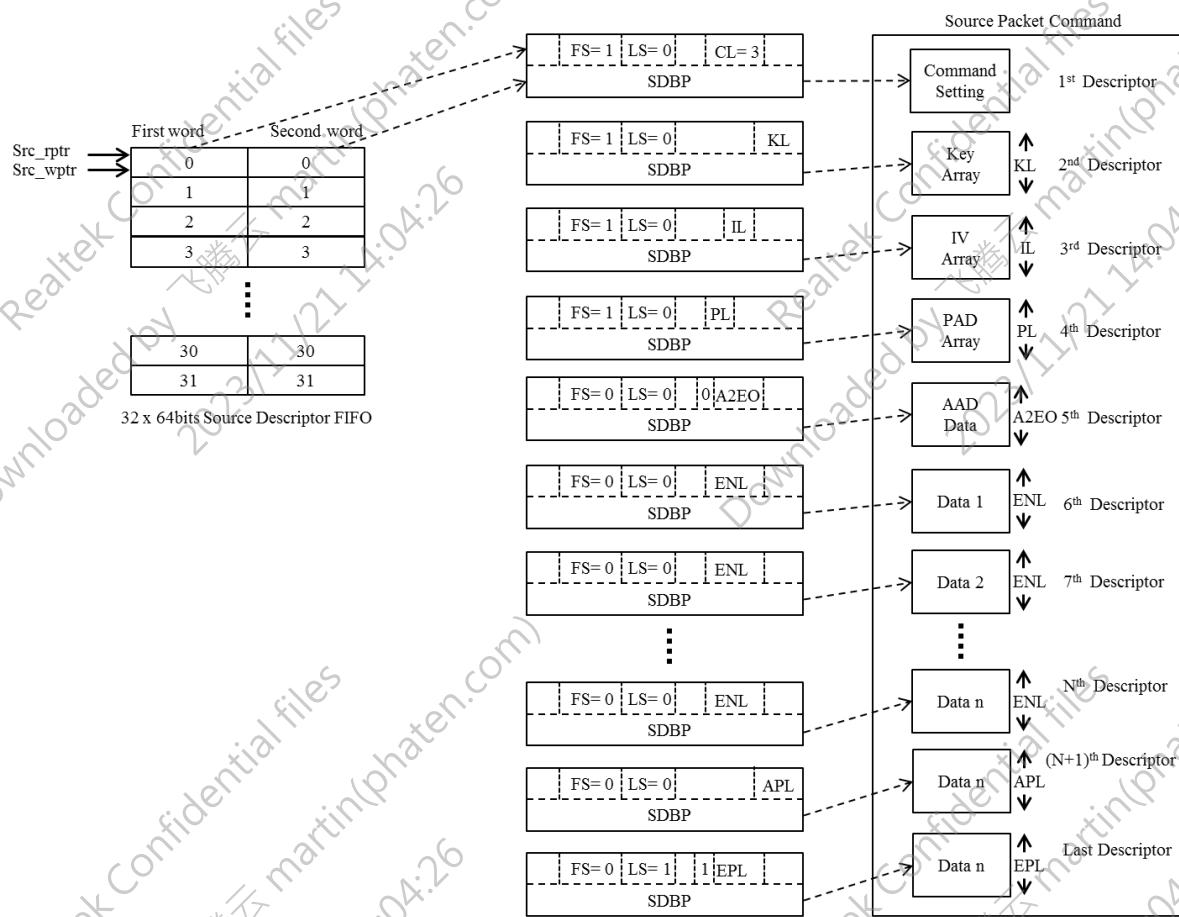


Figure 18-2 Source Descriptor FIFO

18.2.2.1.2 Destination Descriptor FIFO

Destination Descriptor FIFO gets 16 FIFO nodes. Each FIFO node can point to 2 words (32bits) that are First word and Second word. Use First word to set header information and Second word to point to the address of data buffer. Destination Descriptor FIFO uses 2 hardware pointers which are read pointer and write pointer to maintain this FIFO.

Users can set Crypto engine to put the calculated cryptographic feature result into Destination descriptors via setting a Destination Packet Command. A Destination Packet command starts from a descriptor which *FS*=1, then ends this packet in a descriptor which *LS*=1. When set a Destination Packet command, the *ENC* field is very important, it decide this descriptor is set for Cipher result or Authentication digest. The setting sequence must be Cipher first then Authentication. If user only sets for Authentication, then ignore cipher setting.

- **Crypto Data:** *ENL* uses 1byte as a unit.
- **Hash Data:** *ADL* uses 1byte as a unit.

These concepts are illustrated in below:

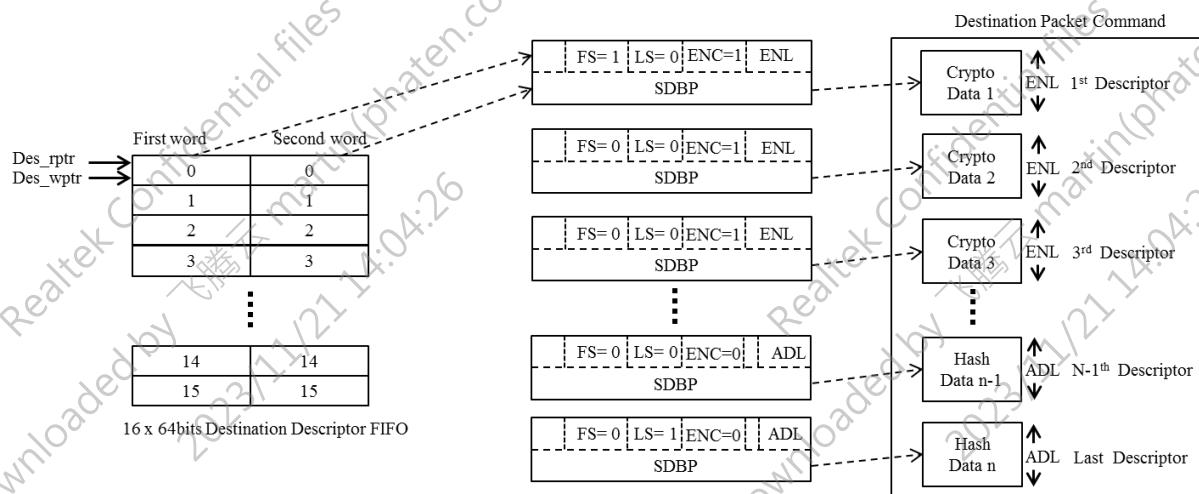


Figure 18-3 Destination Descriptor FIFO

<Note>

User's setting descriptor sequence needs to follow the rule which is setting destination descriptor first and then setting source descriptor.

18.2.2.1.3 Key Array Element

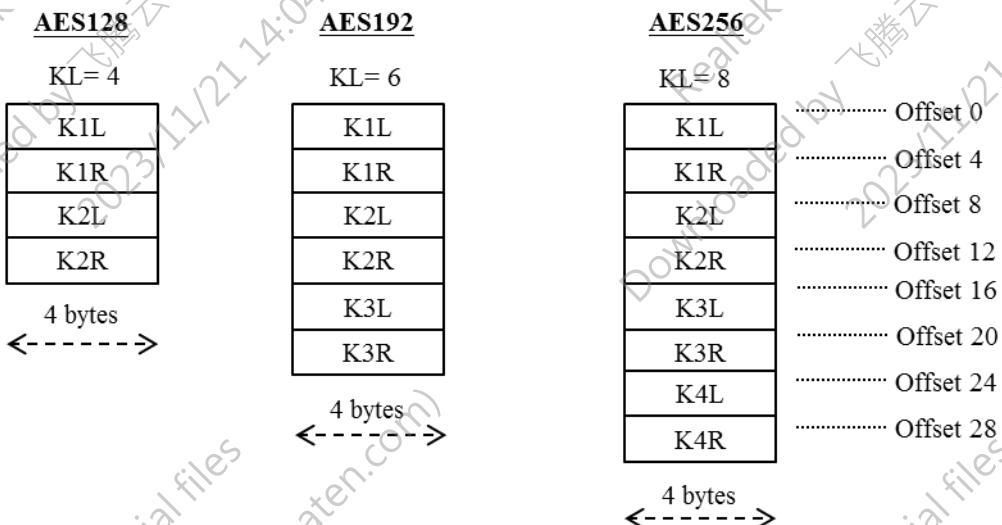


Figure 18-4 Key Array Element

- If Cipher_key_word_swap == 1'b0:

When AES128 : Key = {K1L, K1R, K2L, K2R}.

When AES192 : Key = {K1L, K1R, K2L, K2R, K3L, K3R}.

When AES256 : Key = {K1L, K1R, K2L, K2R, K3L, K3R, K4L, K4R}.

- If Cipher_key_word_swap == 1'b:

When AES128 : Key = {K2R, K2L, K1R, K1L}.

When AES192 : Key = {K3R,K3L,K2R,K2L,K1R,K1L}.

When AES256 : Key = {K4R,K4L,K3R,K3L,K2R,K2L,K1R,K1L}.

18.2.2.1.4 IV Array Element

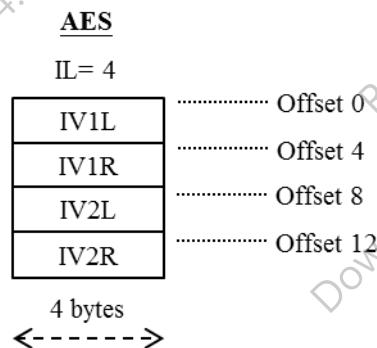


Figure 18-5 IV Array Element

- If Cipher_key_word_swap == 1'b0:

When AES : IV = {IV1L,IV1R,IV2L,IV2R}.

When AES-GCM : nonce = {IV1L,IV1R,IV2L}, counter = {IV2R}.

- If Cipher_key_word_swap == 1'b1:

When AES : IV = {IV2R,IV2L,IV1R,IV1L}.

When AES-GCM : nonce = {IV2R,IV2L,IV1R}, counter = {IV1L}.

18.2.2.1.5 HMAC Key PAD Array Element

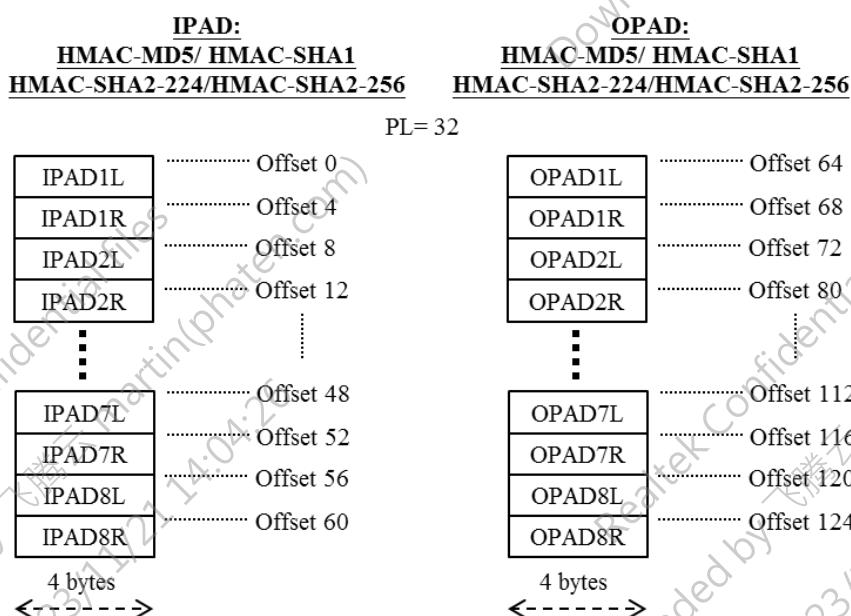


Figure 18-6 HMAC Key PAD Array Element

When HMAC-MD5/HMAC-SHA1/HMAC-SHA2-224/HMAC-SHA2-256:

IPAD = {IPAD1L, IPAD1R, IPAD2L, IPAD2R, ..., IPAD7L, IPAD7R, IPAD8L, IPAD8R}.

OPAD = {OPAD1L, OPAD1R, OPAD2L, OPAD2R, ..., OPAD7L, OPAD7R, OPAD8L, OPAD8R}.

18.2.3 Descriptor Data Structures

The descriptor registers (SDFWR/SDSWR/DDFWR/DDSWR) are pointers. User fills a memory address which contains data to one of them. Crypto engine will follow the descriptor configuration to execute the crypto algorithms. Because there are different configurations in one descriptor register, user can follow below tables know how to set them.

18.2.3.1 Source Crypto Descriptor Type1

This descriptor data type is used to set command setting or other initial values that crypto algorithms need. If user wants to set command setting, need to follow 18.2.3.2 Source Crypto Descriptor Command Setting.

If user wants to set IV(initial value) for cipher algorithms, need to fill IV length to first word(offset 0) and then fill the address of IV data buffer to second word(offset 4).

Table 18-1 SCD_DT1_Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																				
R S V D	R S S D	F LS R S	CL S V D	AP	Hash Initial Value Length, SHIVL (6 bits)				HMAC Key PAD Length, PL (8bits)				IV Length, IL (4 bits)				Key Length, KL (4 bits)				Offset 0																														
Source Data Buffer Pointer, SDBP																													Offset 4																						

Table 18-2 SCD_DT1_Description

Offset#	Bit#	Symbol	Description
0	31-30	RSVD	Reserved bits.
0	29	FS	First segment descriptor: When this field is set "1" and used with <i>CL</i> = "3", it indicates that this is the first descriptor of an IP packet, and this descriptor is pointing to the first segment of the packet. Except setting message buffer address to descriptor, other setting situations (Key Array, IV Array, PAD Array, SHIVL) need set this field is "1" as well.
0	28	LS	Last segment descriptor: When set this field, it indicates that this is the last descriptor of an IP packet, and this descriptor is pointing to the last segment of the packet.

0	27-26	RSVD	Reserved bits.								
0	25-24	CL	Command length: When this field is set “3” and used with <i>FS</i> = “1”, it indicates that this descriptor is the first descriptor of an IP packet and pointing to the command setting buffer address. Other situations will set this field is “0”.								
0	23-22	AP	Auto-padding <table border="1"><tr><td>Value</td><td>Meaning</td></tr><tr><td>00</td><td>Disable Auto-padding(Note: There is no auto-padding in Mix_mode.)</td></tr><tr><td>01</td><td>Auto-padding for MD5/SHA1/SHA2-224/256</td></tr><tr><td>1x</td><td>Auto-padding for SHA2-384/512</td></tr></table>	Value	Meaning	00	Disable Auto-padding(Note: There is no auto-padding in Mix_mode.)	01	Auto-padding for MD5/SHA1/SHA2-224/256	1x	Auto-padding for SHA2-384/512
Value	Meaning										
00	Disable Auto-padding(Note: There is no auto-padding in Mix_mode.)										
01	Auto-padding for MD5/SHA1/SHA2-224/256										
1x	Auto-padding for SHA2-384/512										
0	21-16	SHIVL	Sequential Hash Initial Value Length: This is the length of Sequential Hash Initial Value which uses 4bytes as an unit. If authentication algorithms which use sequential hash mechanism, it may set Sequential Hash Initial Value and Length.								
0	15-8	PL	HMAC Key Pad Length: This is the length of HMAC Key Pad which uses 4byte as a unit for authentication algorithms.								
0	7-4	IL	Initial Vector Length: This is the length of IV which uses 4bytes as a unit for cipher algorithms.								
0	3-0	KL	Key Length: This is the length of Key which uses 4bytes as a unit for cipher algorithms.								
4	31-0	SDBP	Source Data Buffer Pointer: This pointer points to the physical address of source data buffer.								

18.2.3.2 Source Crypto Descriptor Command Setting

If user sets *FS*=1 and *CL*=3 in Source Crypto Descriptor Type1, need to set command setting buffer, and make SDBP point to this buffer.

Table 18-3 SCD_CS_Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICV Total length			SH	SH	SH	Engine	SH	HMAC			HE	CK	RS	RS	AKS			CE	RS	CES	Cipher mode			Offset 0							
ICVTL (8 bits)			N		F	Mode	L	Mode			BS	V	V		AKS			V			, CM (4 bits)										
H	HK	H	H	H	H	H	H	C	C	C	C	C	C	CA	C	HS	Encrypt	CK	Pad Last	Encrypt Last	AAD Last			Offset 4							
KB	W	O	O	I	I	AB	A	I	O	O	O	I	I	BS	A	I	Pad Last	W	Data Size	Data Size,											
S	S	BS	W	BS	W	S	W	D	D	BS	W	BS	W		W	BS	Data Size	S	,PLDS	ELDS	(4 bits)			AADLDS (4 bits)							
Encryption Total Length, ETL (16 bits)															E	Header Total Length, HTL (6 bits)					Hash Padding Total Length, HPTL (8 bits)					Offset 8					
															P																
															T																
															L																

All Message Length1, APL1 (32 bits)	Offset C
All Message Length2, APL2 (32 bits)	Offset 10

Table 18-4 SCD_CS_Description

Offset#	Bit#	Symbol	Description												
0	31-24	ICVTL	Initial Check Vector Total Length (Unit = 1 byte). Set hash digest length in this field. In normal mode(only hash or only cipher) or mix mode(ssh/esp), user usually can assign maximum hash digest length(64bytes = 0x40) in this field. However, when user sets this field for mix mode(ssl_tls_enc), it's essential to set the correct hash digest length.												
0	23	SHNW	Sequential Hash No Write Back: <table border="1" style="margin-left: 20px;"> <tr> <th>Value</th> <th>Meaning</th> </tr> <tr> <td>0</td> <td>Sequential Hash No Write Back Disable. It means hash out the digest.</td> </tr> <tr> <td>1</td> <td>Sequential Hash No Write Back Enable. It means don't hash out the digest.</td> </tr> </table> Sequential Hash references Figure 18-7.	Value	Meaning	0	Sequential Hash No Write Back Disable. It means hash out the digest.	1	Sequential Hash No Write Back Enable. It means don't hash out the digest.						
Value	Meaning														
0	Sequential Hash No Write Back Disable. It means hash out the digest.														
1	Sequential Hash No Write Back Enable. It means don't hash out the digest.														
0	22	SH	Sequential Hash: <table border="1" style="margin-left: 20px;"> <tr> <th>Value</th> <th>Meaning</th> </tr> <tr> <td>0</td> <td>Sequential Hash Mechanism Disable</td> </tr> <tr> <td>1</td> <td>Sequential Hash Mechanism Enable</td> </tr> </table> When programmer sets sequential hash message buffer, it always needs to set "1" to this bit. Sequential Hash references Figure 18-8.	Value	Meaning	0	Sequential Hash Mechanism Disable	1	Sequential Hash Mechanism Enable						
Value	Meaning														
0	Sequential Hash Mechanism Disable														
1	Sequential Hash Mechanism Enable														
0	21	SHF	Sequential Hash First: <table border="1" style="margin-left: 20px;"> <tr> <th>Value</th> <th>Meaning</th> </tr> <tr> <td>0</td> <td>This message buffer isn't sequential hash first one</td> </tr> <tr> <td>1</td> <td>This message buffer is sequential hash first one</td> </tr> </table> Sequential Hash references Figure 18-9.	Value	Meaning	0	This message buffer isn't sequential hash first one	1	This message buffer is sequential hash first one						
Value	Meaning														
0	This message buffer isn't sequential hash first one														
1	This message buffer is sequential hash first one														
0	20-18	EM	Engine Mode: <table border="1" style="margin-left: 20px;"> <tr> <th>Value</th> <th>Meaning</th> </tr> <tr> <td>000</td> <td>Set Crypto engine is Cipher mode</td> </tr> <tr> <td>001</td> <td>Set Crypto engine is Hash mode</td> </tr> <tr> <td>010</td> <td>Set Crypto engine is Mix mode(SSH-enc/ESP-dec)</td> </tr> <tr> <td>011</td> <td>Set Crypto engine is Mix mode(SSH-dec/ESP-enc)</td> </tr> <tr> <td>100</td> <td>Set Crypto engine is Mix mode(SSL_TLS-enc)</td> </tr> </table>	Value	Meaning	000	Set Crypto engine is Cipher mode	001	Set Crypto engine is Hash mode	010	Set Crypto engine is Mix mode(SSH-enc/ESP-dec)	011	Set Crypto engine is Mix mode(SSH-dec/ESP-enc)	100	Set Crypto engine is Mix mode(SSL_TLS-enc)
Value	Meaning														
000	Set Crypto engine is Cipher mode														
001	Set Crypto engine is Hash mode														
010	Set Crypto engine is Mix mode(SSH-enc/ESP-dec)														
011	Set Crypto engine is Mix mode(SSH-dec/ESP-enc)														
100	Set Crypto engine is Mix mode(SSL_TLS-enc)														
0	17	SHL	Sequential Hash Last: <table border="1" style="margin-left: 20px;"> <tr> <th>Value</th> <th>Meaning</th> </tr> <tr> <td>0</td> <td>This message buffer isn't sequential hash last one</td> </tr> <tr> <td>1</td> <td>This message buffer is sequential hash last one</td> </tr> </table> Sequential Hash references Figure 18-10 .	Value	Meaning	0	This message buffer isn't sequential hash last one	1	This message buffer is sequential hash last one						
Value	Meaning														
0	This message buffer isn't sequential hash last one														
1	This message buffer is sequential hash last one														
0	16-14	HM	HMAC Mode: <table border="1" style="margin-left: 20px;"> <tr> <th>Value</th> <th>Meaning</th> </tr> <tr> <td>000</td> <td>MD5</td> </tr> <tr> <td>001</td> <td>SHA1</td> </tr> <tr> <td>010</td> <td>SHA2-224</td> </tr> <tr> <td>011</td> <td>SHA2-256</td> </tr> </table>	Value	Meaning	000	MD5	001	SHA1	010	SHA2-224	011	SHA2-256		
Value	Meaning														
000	MD5														
001	SHA1														
010	SHA2-224														
011	SHA2-256														
0	13	HE	HMAC Enable: <table border="1" style="margin-left: 20px;"> <tr> <th>Value</th> <th>Meaning</th> </tr> <tr> <td>0</td> <td>HMAC Disable</td> </tr> <tr> <td>1</td> <td>HMAC Enable</td> </tr> </table>	Value	Meaning	0	HMAC Disable	1	HMAC Enable						
Value	Meaning														
0	HMAC Disable														
1	HMAC Enable														

0	12	CKBS	Cipher Key Byte Swap:																							
			<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Disable.</td></tr> <tr> <td>1</td><td>Enable</td></tr> </tbody> </table>	Value	Meaning	0	Disable.	1	Enable																	
Value	Meaning																									
0	Disable.																									
1	Enable																									
0	11	RSVD	Reserved bits																							
0	10	RSVD	Reserved bits																							
0	9-8	AKS	AES Key Type Select:																							
			<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>00</td><td>Size of key is 128 bits</td></tr> <tr> <td>01</td><td>Size of key is 192 bits</td></tr> <tr> <td>10</td><td>Size of key is 256 bits</td></tr> </tbody> </table>	Value	Meaning	00	Size of key is 128 bits	01	Size of key is 192 bits	10	Size of key is 256 bits															
Value	Meaning																									
00	Size of key is 128 bits																									
01	Size of key is 192 bits																									
10	Size of key is 256 bits																									
0	7	CE	Cipher Encrypt:																							
			<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Cipher Decrypt</td></tr> <tr> <td>1</td><td>Cipher Encrypt</td></tr> </tbody> </table>	Value	Meaning	0	Cipher Decrypt	1	Cipher Encrypt																	
Value	Meaning																									
0	Cipher Decrypt																									
1	Cipher Encrypt																									
0	6	RSVD	Reserved bits																							
0	5-4	CES	Cipher engine select:																							
			<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>00</td><td>AES</td></tr> </tbody> </table>	Value	Meaning	00	AES																			
Value	Meaning																									
00	AES																									
0	3-0	CM	Cipher mode:																							
			<table border="1"> <thead> <tr> <th>Type</th><th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td rowspan="10">AES Engine</td><td>0000</td><td>ECB mode</td></tr> <tr> <td>0001</td><td>CBC mode</td></tr> <tr> <td>0010</td><td>CFB mode</td></tr> <tr> <td>0011</td><td>OFB mode</td></tr> <tr> <td>0100</td><td>CTR mode</td></tr> <tr> <td>0101</td><td>GCTR mode</td></tr> <tr> <td>0110</td><td>GMAC mode</td></tr> <tr> <td>0111</td><td>GHASH mode</td></tr> <tr> <td>1000</td><td>GCM mode</td></tr> <tr> <td>1001</td><td>AES_CTR32 mode</td></tr> </tbody> </table>	Type	Value	Meaning	AES Engine	0000	ECB mode	0001	CBC mode	0010	CFB mode	0011	OFB mode	0100	CTR mode	0101	GCTR mode	0110	GMAC mode	0111	GHASH mode	1000	GCM mode	1001
Type	Value	Meaning																								
AES Engine	0000	ECB mode																								
	0001	CBC mode																								
	0010	CFB mode																								
	0011	OFB mode																								
	0100	CTR mode																								
	0101	GCTR mode																								
	0110	GMAC mode																								
	0111	GHASH mode																								
	1000	GCM mode																								
	1001	AES_CTR32 mode																								
4	31	HKBS	Hash Key Byte Swap:																							
			<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Disable</td></tr> <tr> <td>1</td><td>Enable</td></tr> </tbody> </table>	Value	Meaning	0	Disable	1	Enable																	
Value	Meaning																									
0	Disable																									
1	Enable																									
4	30	HKWS	Hash Key Word Swap:																							
			<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Disable</td></tr> <tr> <td>1</td><td>Enable</td></tr> </tbody> </table>	Value	Meaning	0	Disable	1	Enable																	
Value	Meaning																									
0	Disable																									
1	Enable																									
4	29	HOBS	Hash Output Byte Swap:																							
			<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Disable</td></tr> <tr> <td>1</td><td>Enable</td></tr> </tbody> </table>	Value	Meaning	0	Disable	1	Enable																	
Value	Meaning																									
0	Disable																									
1	Enable																									
4	28	HOWS	Hash Output Word Swap:																							
			<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Disable</td></tr> <tr> <td>1</td><td>Enable</td></tr> </tbody> </table>	Value	Meaning	0	Disable	1	Enable																	
Value	Meaning																									
0	Disable																									
1	Enable																									
4	27	HIBS	Hash Input Byte Swap:																							
			<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Disable</td></tr> <tr> <td>1</td><td>Enable</td></tr> </tbody> </table>	Value	Meaning	0	Disable	1	Enable																	
Value	Meaning																									
0	Disable																									
1	Enable																									
4	26	HIWS	Hash Input Word Swap:																							
			<table border="1"> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> </table>	Value	Meaning																					
Value	Meaning																									

			<table border="1"> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	0	Disable	1	Enable			
0	Disable									
1	Enable									
4	25	HABS	<p>Hash Aligned Byte Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	24	HAWS	<p>Hash Aligned Word Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	23	CIDWS	<p>Cipher Input Double Word Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	22	CODWS	<p>Cipher Output Double Word Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	21	COBS	<p>Cipher Output Byte Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	20	COWS	<p>Cipher Output Word Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	19	CIBS	<p>Cipher Input Byte Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	18	CIWS	<p>Cipher Input Word Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	17	CABS	<p>Cipher Aligned Byte Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	16	CAWS	<p>Cipher Aligned Word Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	15	HSIBS	<p>Hash Sequential Initial Byte Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	14-12	EPLDS	Encryption Padding Last Data Size.(For Mix mode:SSL_TLS-enc)							
4	11	CKWS	<p>Cipher Key Word Swap:</p> <table border="1"> <tr><td>Value</td><td>Meaning</td></tr> <tr><td>0</td><td>Disable</td></tr> <tr><td>1</td><td>Enable</td></tr> </table>	Value	Meaning	0	Disable	1	Enable	
Value	Meaning									
0	Disable									
1	Enable									
4	10-8	PLDS	Hash Padding Last Data size.(For mix mode Hash padding)							
4	7-4	ELDS	Encryption Last Data Size:							

			Record the last data size of the message which will be encrypted, no matter whether the last data size is aligned or not.										
4	3-0	AADLDS	AAD Last Data Size: Record the AAD last data size, no matter whether the AAD last data size is aligned or not.										
8	31-16	ETL	Encryption Total Length (Cipher/Hash): This is the total length of message buffer that Crypto engine can calculate. A unit for different cryptographic features has different length meanings. The details are listed below: <table border="1" data-bbox="504 458 1426 631"> <thead> <tr> <th>Type</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>Hash (MD5/SHA1/SHA2-224/SHA2-256)</td><td>64 bytes as a unit</td></tr> <tr> <td>Hash (MD5/SHA1/SHA2-224/SHA2-256) Auto Padding</td><td>16 bytes as a unit</td></tr> <tr> <td>Cipher (AES)</td><td>16 bytes as a unit</td></tr> <tr> <td>Mix mode SSH/ESP/SSL-TLS</td><td>16 bytes as a unit</td></tr> </tbody> </table>	Type	Meaning	Hash (MD5/SHA1/SHA2-224/SHA2-256)	64 bytes as a unit	Hash (MD5/SHA1/SHA2-224/SHA2-256) Auto Padding	16 bytes as a unit	Cipher (AES)	16 bytes as a unit	Mix mode SSH/ESP/SSL-TLS	16 bytes as a unit
Type	Meaning												
Hash (MD5/SHA1/SHA2-224/SHA2-256)	64 bytes as a unit												
Hash (MD5/SHA1/SHA2-224/SHA2-256) Auto Padding	16 bytes as a unit												
Cipher (AES)	16 bytes as a unit												
Mix mode SSH/ESP/SSL-TLS	16 bytes as a unit												
8	15-14	EPTL	Encryption Padding Total Length. (For Mix mode: SSL_TLS-enc) It uses 8bytes as a unit.										
8	13-8	HTL	Header Total Length (AAD total length). This is the total length of AAD data. For AES_GCM, it uses 16bytes as a unit. However, for mix mode (SSH/ESP/SSL_TLS), they use 8bytes as a unit										
8	7-0	HPTL	Hash Padding Total Length. (For Mix mode: SSH/ESP/SSL_TLS) They use 8bytes as a unit. Because mix mode doesn't support hash auto-padding, so user needs to set this field for hash padding.										
12	31-0	APL1	All Message Length1 For Auto Padding Information. This is usually to record all hash message length in bits. If the length size is more than APL1 range ($2^{32} - 1$), then may use APL2.										
14	31-0	APL2	All Message Length2 For Auto Padding Information. This is usually to record all hash message length in bits. The length maximum size is APL1 and APL2 total length range ($2^{64} - 1$).										

Sequential Hash Mechanism:

If a message payload buffer size is too large for Crypto engine to handles, then split the buffer into many 64byte-aligned size message payload. As for the last one message payload, its size must be 16byte-aligned before Crypto engine handling, so if the last one size isn't 16byte-aligned, need to fill suffix padding values. In the end, if you set auto-padding filed in source descriptor buffer, then Crypto engine will auto-padding for this long hash message payload, then generate(hash out) the digest.

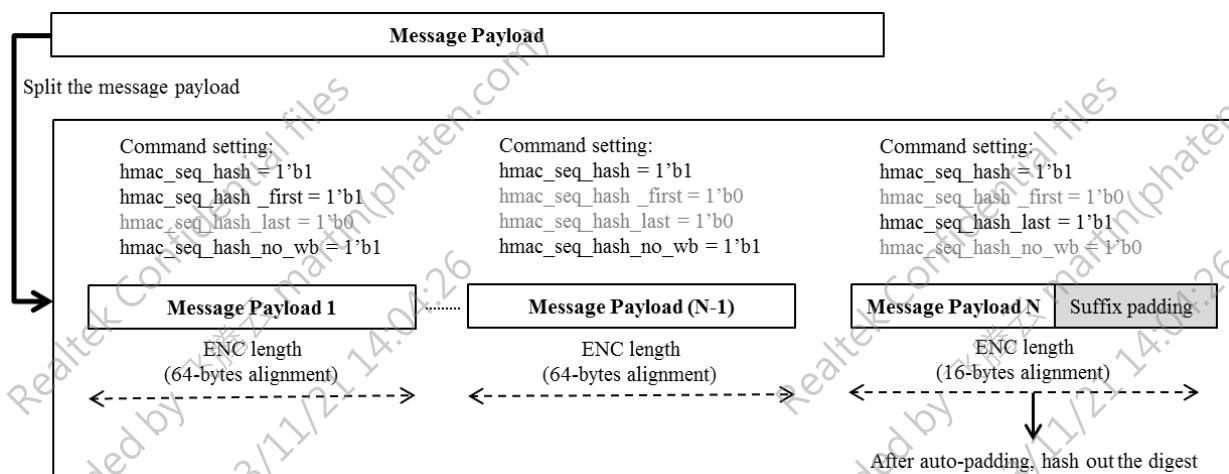


Figure 18-11 Sequential Hash Mechanism

18.2.3.3 Source Crypto Descriptor Type2

This descriptor data type is used to set data buffer that Crypto engine will process.

If user wants to set message buffer for cipher algorithms, need to fill Encryption Length(ENL) to first word(offset 0) and then fill the address of message buffer to second word(offset 4).

Table 18-5 SCD_DT2_Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	R	F	LS	Encryption Length, ENL (14 bits)														Z	Authentication		Authentication Padding							Offset 0			
S	S																	E	to Encryption		Length, APL										
V	S																	O	Offset, A2EO (5bits)												
D																		Encryption													
Source Data Buffer Pointer, SDBP																												Offset 4			

Table 18-6 SCD_DT2_Description

Offset#	Bit#	Symbol	Description						
0	31-30	RSVD	Reserved bits.						
0	29	FS	First segment descriptor: When this field is set "0", it indicates that this descriptor is used to set message buffer.						
0	28	LS	Last segment descriptor: When set this field, it indicates that this is the last descriptor of an IP packet, and this descriptor is pointing to the last segment of the packet.						
0	27-14	ENL	Encryption Data Length: This is the length of message buffer in byte which Hash/Cipher handling. Length is 1byte alignment, the maximum is 16383 bytes.						
0	13	ZERO	Decide the bit field [12-8] content is A2EO or EPL. <table border="1" data-bbox="504 1500 1389 1594"> <tr> <th>Value</th><th>Meaning</th></tr> <tr> <td>0</td><td>A2EO(Header) data length</td></tr> <tr> <td>1</td><td>EPL (Encryption Padding Length) for Mix mode</td></tr> </table>	Value	Meaning	0	A2EO(Header) data length	1	EPL (Encryption Padding Length) for Mix mode
Value	Meaning								
0	A2EO(Header) data length								
1	EPL (Encryption Padding Length) for Mix mode								
0	12-8	A2EO/EPL	Authentication to Encryption Offset (Additional Authentication Data Length) or EPL(Encryption Padding Length). This field uses 1bytes as a unit, the maximum is 31 bytes. And the length meaning depends on the bit field [13]. A2eo is set for header, EPL is usually set for mix mode(SSL_TLS-enc).						
0	7-0	APL	Authentication(Hash) Padding Length: This field is set to the Hash padding data length. If the field (Auto-padding in Source Descriptor type 1) is set to enable auto padding, then don't need to set this field. Usually, user will set this field for mix mode, because mix mode doesn't support Hash auto-padding mechanism. Length is 1byte alignment, the maximum is 255 bytes.						
4	31-0	SDBP	Source Data Buffer Pointer: This pointer points to the physical address of source data buffer.						

18.2.3.4 Destination Crypto Descriptor Type1

This descriptor data type is setting for Authentication digest after Crypto engine process.

If user wants to set digest buffer for Hash algorithms, need to fill Authentication Data Length(ADL) to first word(offset 0) and then fill the address of digest buffer to second word(offset 4).

Table 18-7 DCD_DT1_Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R S V D	W S S D	F L S C	RSVD	Authentication Data Length, ADL (8 bits)	Offset 0																										
			Destination Data Buffer Pointer, DDBP	Offset 4																											

Table 18-8 DCD_DT1_Description

Offset#	Bit#	Symbol	Description						
0	31-30	RSVD	Reserved bits.						
0	29	FS	First segment descriptor: When this field is set "1", it indicates that this is the first descriptor of an IP packet, and this descriptor is pointing to the first segment of the packet.						
0	28	LS	Last segment descriptor: When set this field, it indicates that this is the last descriptor of an IP packet, and this descriptor is pointing to the last segment of the packet.						
0	27	ENC	Encryption Flag: <table border="1"> <tr> <td>Value</td><td>Meaning</td></tr> <tr> <td>0</td><td>Authentication algorithms</td></tr> <tr> <td>1</td><td>Cipher algorithms</td></tr> </table> Set this field is "0" for Destination Crypto Descriptor Type1.	Value	Meaning	0	Authentication algorithms	1	Cipher algorithms
Value	Meaning								
0	Authentication algorithms								
1	Cipher algorithms								
0	26-8	RSVD	Reserved bits						
0	7-0	ADL	Authentication Data Length: This is the length of digest which uses 1byte as a unit for Hash algorithms and Tag value in cipher algorithms. The maximum is 255 bytes.						
4	31-0	DDBP	Destination Data Buffer Pointer: This pointer points to the physical address of destination data buffer.						

18.2.3.5 Destination Crypto Descriptor Type2

This descriptor data type is setting for Cipher result after Crypto engine process

If user wants to set cipher result buffer for Hash algorithms, need to fill Encryption Length(ENL) to first word(offset 0) and then fill the address of cipher result buffer to second word(offset 4).

Table 18-9 DCD_DT2_Format

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

R	W	F	LS	EN	RSVD	Encryption Length, ENL (24 bits)	Offset 0
V	S	S		C		Destination Data Buffer Pointer, DDBP	Offset 4

Table 18-10 DCD_DT2_Description

Offset#	Bit#	Symbol	Description							
0	31-30	RSVD	Reserved bits.							
0	29	FS	First segment descriptor: When this field is set “1”, it indicates that this is the first descriptor of an IP packet, and this descriptor is pointing to the first segment of the packet.							
0	28	LS	Last segment descriptor: When set this field, it indicates that this is the last descriptor of an IP packet, and this descriptor is pointing to the last segment of the packet.							
0	27	ENC	Encryption Flag: <table border="1"> <tr> <th>Value</th><th>Meaning</th></tr> <tr> <td>0</td><td>Authentication algorithms</td></tr> <tr> <td>1</td><td>Cipher algorithms</td></tr> </table> Set this field is “1” for Destination Crypto Descriptor Type2.	Value	Meaning	0	Authentication algorithms	1	Cipher algorithms	
Value	Meaning									
0	Authentication algorithms									
1	Cipher algorithms									
0	26-24	RSVD	Reserved bits							
0	23-0	ENL	Encryption Length: This is the length of cipher result which uses 1byte as a unit for cipher algorithms. The maximum is 16777215 bytes.							
4	31-0	DDBP	Destination Data Buffer Pointer: This pointer points to the physical address of destination data buffer.							

18.3 Control Registers

18.3.1 SDSR

- Name: Source Descriptor Status Register
- Width: 32bit
- Initial Value: 0x40000000
- Note: This register is used to check Source Descriptor FIFO status.

REG 18-11 (Offset 0000h) SDSR

Bit	Access	INI	Symbol	Description
31	R	0	RSVD	
30	R/W	1	PK_UP	Packet base update wptr to engine. If PK_UP =1, the total number of source descriptor in one packet can't be over 32, and the total number of destination descriptor

				in one packet can't be over 16.
29:28	R	0	RSVD	
27	R/W	0	SRC_FAIL_M	Source Descriptor failed interrupt mask 1'd0: Disable mask 1'd1: Enable mask
26:25	R/W	0	SRC_FAIL_STATUS	Source Descriptor failed status. 2'd1: Users write SDFW twice consecutively. 2'd2: Users write SDSW directly without writing SDFW in the beginning. 2'd3: Overflow (Detect users try to write source descriptor, but there isn't available FIFO node could use).
24	R/W	0	SRC_FAIL	Source Descriptor failed interrupt. Write 1 to clear this bit.
23:16	R	0	SRPTR	Source Descriptor FIFO read pointer. When engine read a descriptor and finished it, SRPTR = SRPTR+2
15:8	R	0	SWPTR	Source Descriptor FIFO write pointer. When write descriptor to SDSW successfully, SWPTR = SWPTR+2.
7:0	R	0	FIFO_EMPTY_CNT	Source Descriptor FIFO empty counter. Use this field to check how many available FIFO nodes in Source descriptor FIFO could use.

18.3.2 SDFWR

- Name: Source Descriptor First Word Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to set descriptor header. The programmer can follow Source Crypto Descriptor Type1/2 offset 0 to set header information data, and write this data to this register.

REG 18-12 (Offset 0004h) SDFWR

Bit	Access	INI	Symbol	Description
31:0	R/W	0	SDFW	Source Descriptor First Word

18.3.3 SDSWR

- Name: Source Descriptor Second Word Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to set descriptor data buffer address. The programmer can know this register is a hardware pointer which records the data buffer address from Source Crypto Descriptor Type1/2 offset 4. If the programmer sets a descriptor, need to write the header information data in first word register and write data buffer address in

second word register.

REG 18-13 (Offset 0008h) SDSWR

Bit	Access	INI	Symbol	Description
31:0	R/W	0	SDSW	Source Descriptor Second Word

18.3.4 IPCSR_CONF_INT

- Name: Crypto Engine Command/Status Configure Interrupt Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register allows the programmer can reset crypto engine and DMA engine, check and configure crypto engine interrupt status.

REG 18-14 (Offset 0010h) IPCSR_CONF_INT

Bit	Access	INI	Symbol	Description
31	W	0	IPSEC_RST	Crypto engine reset. Write "1" to reset the crypto engine and DMA engine. (Use to clear fatal error)
30:24	R	0	RSVD	
23:16	R/W	0	CLEAR_OK_INT_NUM	Clear ok interrupt number. In interrupt counter mode, if the programmer wants to clear <i>OK_INTR_CNT</i> , need to write <i>OK_INTR_CNT</i> to this field first, then write "1" to <i>CMD_OK</i> .
15:8	R	0	OK_INTR_CNT	Ok interrupt counter. Read this field can know how many interrupts are coming to notify crypto engine has calculated cryptographic feature results. Use this field in interrupt counter mode.
7	R/W	0	INTR_MODE	Select ok interrupt mode: 1'd0: Interrupt mode is general mode. 1'd1: Interrupt mode is counter mode. General mode means no matter how many interrupts are coming to notify crypto engine has calculated cryptographic feature results, there always is one interrupt signal shows in <i>CMD_OK</i> field, and nothing in <i>OK_INTR_CNT</i> . So if the programmer want to clear this signal, just write "1" to <i>CMD_OK</i> . Counter mode means the programmer can know how many interrupts are coming to notify crypto engine has calculated cryptographic feature results from <i>OK_INTR_CNT</i> . The programmer also can know there are at least one interrupt from reading <i>CMD_OK</i> . So if the programmer want to clear <i>OK_INTR_CNT</i> signals, need to write <i>OK_INTR_CNT</i> into <i>CLEAR_OK_INT_NUM</i> , then write "1" to <i>CMD_ok</i> .

6:5	R	0	RSVD	
4	R/W	0	CMD_OK	Command ok Interrupt. Read this field to detect whether crypto engine has calculated a cryptographic feature result. Even if interrupt mode is counter mode, still can use this field to detect whether an interrupt is coming. Write "1" to clear this interrupt signal.
3	R	0	DMA_BUSY	Detect whether Crypto engine DMA is busy: 1'd0: DMA is not busy 1'd1: DMA is busy Using for debugging. To avoid data coherence issue[when user get crypto finish calculated interrupt, user can make sure DMA engine has already move data to result buffer.]
2:1	R	0	RSVD	
0	W	0	SOFT_RST	Software Reset. Write "1" to reset the crypto engine.

18.3.5 IPCSR_INT_MASK

- Name: Crypto Engine Command/Status Interrupt Mask Control Register
- Width: 32bit
- Initial Value: 0x0007FFF9
- Note: This register is used to enable or disable Command ok interrupt mask, Source descriptor error interrupt mask and Destination descriptor error interrupt mask.

REG 18-15 (Offset 0014h) IPCSR_INT_MASK

Bit	Access	INI	Symbol	Description
31:19	R	0	RSVD	
18	R/W	0	DES_ERR5_M	Destination Descriptor Error 5 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
17	R/W	0	DES_ERR4_M	Destination Descriptor Error 4 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
16	R/W	0	DES_ERR3_M	Destination Descriptor Error 3 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
15	R/W	0	DES_ERR2_M	Destination Descriptor Error 2 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
14	R/W	0	DES_ERR1_M	Destination Descriptor Error 1 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
13	R/W	1	DES_ERR0_M	Destination Descriptor Error 0 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
12	R/W	1	SRC_ERR9_M	Source Descriptor Error 9 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
11	R/W	1	SRC_ERR8_M	Source Descriptor Error 8 Interrupt Mask

				1'd0: Disable mask 1'd1: Enable mask
10	R/W	1	SRC_ERR7_M	Source Descriptor Error 7 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
9	R/W	1	SRC_ERR6_M	Source Descriptor Error 6 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
8	R/W	1	SRC_ERR5_M	Source Descriptor Error 5 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
7	R/W	1	SRC_ERR4_M	Source Descriptor Error 4 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
6	R/W	1	SRC_ERR3_M	Source Descriptor Error 3 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
5	R/W	1	SRC_ERR2_M	Source Descriptor Error 2 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
4	R/W	1	SRC_ERR1_M	Source Descriptor Error 1 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
3	R/W	1	SRC_ERR0_M	Source Descriptor Error 0 Interrupt Mask 1'd0: Disable mask 1'd1: Enable mask
2:1	R	0	RSVD	
0	R/W	1	CMD_OK_M	Command ok Interrupt Mask. 1'd0: Disable mask 1'd1: Enable mask If the programmer wants to disable <i>cmd_ok</i> interrupt signal, write "1" to enable interrupt mask.

18.3.6 IPCSR_DBG_DMA

- Name: Crypto Engine Command/Status Debug DMA Register
- Width: 32bit
- Initial Value: 0x0100
- Note: This register is used to configure or debug Crypto DMA engine.

REG 18-16 (Offset 0018h) IPCSR_DBG_DMA

Bit	Access	INI	Symbol	Description
31	R/W	0	DEBUG_WB	Debug: write back mode. 1'd0: Disable DMA write back mode. 1'd1: Enable DMA write back mode. Enable this field, Crypto DMA will move data from source address to destination address. During this period, the data won't be processed in any calculation(Encryption, Decryption, Hash). Then the programmer can check whether the source data is the same as destination data.

30:25	R	0	RSVD	
24	R/W	0	ENGINE_CLK_EN	Crypto Engine clock enable 1'd0: Disable Crypto engine clock 1'd1: Enable Crypto engine clock
23:20	R/W	0	DEBUG_PORT_SEL	Debug port selection: 4'd0: engine_debug 4'd1: dma_ixera_debug 4'd2: dma_rx_debug 4'd3: dma_tx_debug
19:17	R	0	RSVD	
16	R/W	1	ARBITER_MODE	DMA arbiter mode: 1'd0: round-robin 1'd1: detect fifo wasted level There are 2 fifo in DMA, that are read-fifo and write-fifo. Detect fifo wasted level means that DMA will process which fifo first depend on which wasted level is high. Because the wasted level is high means there are many fifo nodes waiting to be processed in this fifo, so DMA must process them in hurry.
15:0	R/W	0	DMA_WAIT_CYCLE	Set DMA wait cycles to assert next DMA request.

18.3.7 IPCSR_ERR_STATUS

- Name: Crypto Engine Command/Status Error Status Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to detect Source/Destination descriptor setting error interrupts.

REG 18-17 (Offset 001Ch) IPCSR_ERR_STATUS

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15	R/W	0	DES_ERR5	Destination Descriptor Error 5 Interrupt. The error condition is authentication digest length error in destination descriptor. When set sequential hash: MD5: adl != 16 or SHA1: adl != 20 or SHA2-224 adl != 28 or SHA2-256 adl != 32 or SHA2-384 adl != 48 or SHA2-512 adl != 64 Write "1" to clear this error signal.
14	R/W	0	DES_ERR4	Destination Descriptor Error 4 Interrupt. The error condition is authentication digest length error in destination descriptor. MD5: adl > 16 or SHA1: adl > 20 or SHA2-224 adl > 28 or SHA2-256 adl > 32 or SHA2-384 adl > 48 or

				SHA2-512 adl > 64 Write "1" to clear this error signal.
13	R/W	0	DES_ERR3	Destination Descriptor Error 3 Interrupt. One of the error condition is the sum of <i>enl</i> is greater than Encryption Total Length. The other error condition is about destination descriptor length error ((<i>enl</i> /Encryption length)==0) & (<i>adl</i> (authentication data length)==0)). Write "1" to clear this error signal. Sum of <i>enl</i> reference the register <i>REG_IPCSR_SENLR</i> (offset 0x24).
12	R/W	0	DES_ERR2	Destination Descriptor Error 2 Interrupt. The error condition is the bit field <i>ENC</i> =1'b0(active Hash engine to process hash data), but only active Cipher engine. Write "1" to clear this error signal.
11	R/W	0	DES_ERR1	Destination Descriptor Error 1 Interrupt. The error condition is the bit field <i>ENC</i> =1'b1(active cipher engine to process cipher data), but only active Hash engine. Write "1" to clear this error signal.
10	R/W	0	DES_ERR0	Destination Descriptor Error 0 Interrupt. One of the error condition is destination descriptor starting address error (DDSA[1:0]!=2'd0 [At least 4 bytes-aligned]). The other error condition is memory protection event. Write "1" to clear this error signal.
9	R/W	0	SRC_ERR9	Source Descriptor Error 9 Interrupt. The error condition is sum of <i>epl</i> is not equal to <i>EPTL</i> (Encryption Padding Total Length). Write "1" to clear this error signal. Sum of <i>epl</i> reference the register <i>REG_IPCSR_SEPLR</i> (offset 0x2C).
8	R/W	0	SRC_ERR8	Source Descriptor Error 8 Interrupt. The error condition is sum of <i>apl</i> is not equal to <i>HPTL</i> (Hash Padding Total Length). Write "1" to clear this error signal. Sum of <i>apl</i> reference the register <i>REG_IPCSR_SAPLR</i> (offset 0x28).
7	R/W	0	SRC_ERR7	Source Descriptor Error 7 Interrupt. The error condition is sum of <i>enl</i> is not equal to <i>ETL</i> (Encryption Total Length). Write "1" to clear this error signal. Sum of <i>enl</i> reference the register <i>REG_IPCSR_SENLR</i> (offset 0x24).
6	R/W	0	SRC_ERR6	Source Descriptor Error 6 Interrupt. The error condition is sum of <i>a2eo</i> is not equal to <i>HTL</i> (Header Total Length). Write "1" to clear this error signal. Sum of <i>a2eo</i> reference the register <i>REG_IPCSR_SAADLR</i> (offset 0x20).
5	R/W	0	SRC_ERR5	Source Descriptor Error 5 Interrupt. The error condition is there is a source descriptor, but the data length(<i>a2eo/epl</i>)+(<i>enl</i>) +(<i>apl</i>) is 0. Write "1" to clear this error signal.
4	R/W	0	SRC_ERR4	Source Descriptor Error 4 Interrupt. The error condition is source data buffer pointer error (SDBP[1:0]!=2'd0[At least 4 byte aligned]). Write "1" to clear this error signal.
3	R/W	0	SRC_ERR3	Source Descriptor Error 3 Interrupt. About source descriptor length error. One of the error condition is set the first descriptor but the (<i>CL</i> !=3). The other error condition is there is a source descriptor but <i>CL</i> , <i>KL</i> , <i>IL</i> , <i>PL</i> , <i>SHIVL</i>

				are all 0. Write "1" to clear this error signal.
2	R/W	0	SRC_ERR2	Source Descriptor Error 2 Interrupt. One of the error condition is Source descriptor starting address error ($SDSA[1:0] \neq 2'd0$ [At least 4 bytes-aligned]). The other error condition is memory protection event. Write "1" to clear this error signal.
1	R/W	0	SRC_ERR1	Source Descriptor Error 1 Interrupt. The error condition is last Segment descriptor is set ($LS=1$), when descriptor is pointing to the first segment of the packet. Write "1" to clear this error signal.
0	R/W	0	SRC_ERR0	Source Descriptor Error 0 Interrupt. One of the error condition is first segment descriptor is not set ($FS=0$), when descriptor is pointing to the first segment of the packet. The other error condition is first segment descriptor is set ($FS=1$), when descriptor is not pointing to the first segment of the packet. Write "1" to clear this error signal.

18.3.8 IPCSR_SUM_OF_AADL

- Name: Crypto Engine Command/Status Sum Of Additional Authentication Data Length Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to get sum of additional authentication data length.

REG 18-18 (Offset 0020h) IPCSR_SUM_OF_AADL

Bit	Access	INI	Symbol	Description
31:11	R	0	RSVD	
10:0	R	0	A2EO_SUM	Sum(a2eo): Sum of additional authentication data length.

18.3.9 IPCSR_SUM_OF_EDL

- Name: Crypto Engine Command/Status Sum Of Encryption Data Length Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to get sum of encryption data length.

REG 18-19 (Offset 0024h) IPCSR_SUM_OF_EDL

Bit	Access	INI	Symbol	Description
31:24	R	0	RSVD	
23:0	R	0	ENL_SUM	Sum(enl): Sum of encryption data length.

18.3.10 IPCSR_SUM_OF_APDL

- Name: Crypto Engine Command/Status Sum Of Authentication Padding Data Length Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to get sum of authentication padding data length.

REG 18-20 (Offset 0028h) IPCSR_SUM_OF_APDL

Bit	Access	INI	Symbol	Description
31:12	R	0	RSVD	
11:0	R	0	APL_SUM	Sum(apl): Sum of authentication padding data length.[Mix mode uses this field]

18.3.11 IPCSR_SUM_OF_EPDL

- Name: Crypto Engine Command/Status Sum Of Encryption Padding Data Length Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to get sum of encryption padding data length.

REG 18-21 (Offset 002Ch) IC_HS_SCL_LCNT

Bit	Access	INI	Symbol	Description
31:6	R	0	RSVD	
5:0	R	0	EPL_SUM	Sum(epl): Sum of encryption padding data length. [Mix mode uses this field].

18.3.12 IPCSR_SWAP_BURST

- Name: Crypto Engine Command/Status Swap And Burst Configure Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register configures swap and endian setting in crypto engine, and DMA burst length.

REG 18-22 (Offset 0030h) IPCSR_SWP_BURST

Bit	Access	INI	Symbol	Description
31:22	R	0	RSVD	
21:16	R/W	0	DMA_BURST_LENGTH	Set DMA burst length: The maximum setting length is 32bytes. The minimum setting length is 1 byte. Note:If all bits are zero(6'd0), it means maximum burst size or undefined burst size.
15:13	R	0	RSVF	
12	R/W	0	TX_WD_SWAP	Word swap for dma_tx engine input data: 1'd0: Disable 1'd1: Enable
11	R/W	0	RX_WD_SWAP	Word swap for dma_rx engine output data: 1'd0: Disable 1'd1: Enable
10	R/W	0	MAC_OUT_LITTLE_ENDIAN	Ouput mac is little endian: 1'd0: Big endian 1'd1: Little endian
9	R/W	0	DATA_OUT_LITTLE_ENDIAN	Ouput data is little endian: 1'd0: Big endian 1'd1: Little endian
8	R/W	0	TX_BYTE_SWAP	Byte swap for dma_tx engine input data: 1'd0: Disable 1'd1: Enable
7:5	R	0	RSVD	
4	R/W	0	DATA_IN_LITTLE_ENDIAN	Input data is little endian: 1'd0: Big endian 1'd1: Little endian
3	R/W	0	HASH_INITIAL_VALUE_SWAP	Byte swap for sequential hash initial value: 1'd0: Disable 1'd1: Enable
2	R/W	0	KEY_PAD_SWAP	Byte swap for HMAC key pad: 1'd0: Disable 1'd1: Enable
1	R/W	0	KEY_IV_SWAP	Byte swap for Cipher key and Initial Vector: 1'd0: Disable 1'd1: Enable
0	R/W	0	SET_SWAP	Byte swap for command setting data: 1'd0: Disable 1'd1: Enable

18.3.13 DDSR

- Name: Destination Descriptor Status Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to check Destination Descriptor FIFO status.

REG 18-23 (Offset 1000h) DDSR

Bit	Access	INI	Symbol	Description
31	R/W	0	DES_RST	Destination descriptor reset
30:28	R	0	RSVD	
27	R/W	0	DES_FAIL_M	Destination Descriptor failed interrupt mask: 1'd0: Disable mask 1'd1: Enable mask
26:25	R	0	DES_FAIL_STATUS	Destination Descriptor failed status: 2'd1: Users write DDFW twice consecutively. 2'd2: Users write DDSW directly without writing DDFW in the beginning. 2'd3: Overflow (Detect users try to write source descriptor, but there isn't available FIFO node could use).
24	R/W	0	DES_FAIL	Destination Descriptor failed interrupt. Write 1 to clear this bit.
23:16	R	0	DRPTR	Destination Descriptor FIFO read pointer. When engine read a descriptor and finished it, DRPTR = DRPTR+2
15:8	R	0	DWPTR	Destination Descriptor FIFO write pointer. When write descriptor to DDSW successfully, DWPTR = DWPTR+2
7:0	R	0	FIFO_EMPTY_CNT	Destination Descriptor FIFO empty counter. Use this field to check how many available FIFO nodes in Destination descriptor FIFO could use.

18.3.14 DDFWR

- Name: Destination Descriptor First Word Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to set descriptor header. The programmer can follow Destination Crypto Descriptor Type1/2 offset 0 to set header information data, and write this data to this register.

REG 18-24 (Offset 1004h) DDFWR

Bit	Access	INI	Symbol	Description
31:0	R/W	0	DDFW	Destination Descriptor First Word

18.3.15 DDSWR

- Name: Destination Descriptor Second Word Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to set descriptor data buffer address. The programmer can know this register is a hardware pointer which records the data buffer address from Destination Crypto Descriptor Type1/2 offset 4. If the programmer sets a descriptor, need

to write the header information data in first word register and write data buffer address in second word register.

REG 18-25 (Offset 1008h) DDSWR

Bit	Access	INI	Symbol	Description
31:0	R/W	0	DDSW	Destination Descriptor Second Word

18.3.16 IPCSR_CONF_PKTABR

- Name: Crypto Engine Command/Status Configure Packet Arbiter Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register is used to configure packet arbiter and debug Source/Destination descriptor FIFO read pointer.

REG 18-26 (Offset 100Ch) IPCSR_CONF_PKTABR

Bit	Access	INI	Symbol	Description
31:16	R	0	RSVD	
15:8	R/W	0	DBG_DPTR	Destination Descriptor FIFO read pointer: Users read this pointer value can know what the ID number of FIFO node that is now pointed by read pointer is. If Users want to know what the 32bits value of a specified FIFO node, users can write an ID number to this read pointer, and then get the 32bits value from <i>Debug Destination Descriptor Data Register</i> .
7:0	R/W	0	DBG_SPTR	Source Descriptor FIFO read pointer: Users read this pointer value can know what the ID number of FIFO node that is now pointed by read pointer is. If Users want to know what the 32bits value of a specified FIFO node, users can write an ID number to this read pointer, and then get the 32bits value from <i>Debug Source Descriptor Data Register</i> .

18.3.17 IPCSR_DBG_SDDR

- Name: Crypto Engine Command/Status Debug Source Descriptor Data Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register helps to debug source descriptor FIFO read pointer data.

REG 18-27 (Offset 1010h) IPCSR_DBG_SDDR

Bit	Access	INI	Symbol	Description
31:0	R	0	DBG_SD	Debug Source Descriptor Data: This register is used with <i>DBG_SPTR</i> .

18.3.18 ICSR_DBG_DDDR

- Name: Crypto Engine Command/Status Debug Destination Descriptor Data Register
- Width: 32bit
- Initial Value: 0x0000
- Note: This register helps to debug destination descriptor FIFO read pointer data.

REG 18-28 (Offset 1014h) IPCSR_DBG_DDDR

Bit	Access	INI	Symbol	Description
31:0	R	0	DBG_DD	Debug Destination Descriptor Data: This register is used with <i>DBG_DPTR</i> .

19 Wi-Fi

19.1 Overview

19.1.1 Application Scenario

The Ameba-Z II Wi-Fi unit is a highly integrated single-chip low power 802.11n Wireless LAN (WLAN) network controller. It combines a Wi-Fi media access control (MAC), a 1T1R capable Wi-Fi baseband, and RF in a single chip.

The Wi-Fi MAC unit supports 802.11e for multimedia applications, 802.11i for security, and 802.11n for enhanced MAC protocol efficiency. Using packet aggregation techniques such as A-MPDU with BA and A-MSDU, protocol efficiency is significantly improved. Power saving mechanisms such as Legacy Power Save, and U-APSD, reduce the power wasted during idle time, and compensate for the extra power required to transmit OFDM.

The Wi-Fi unit built-in enhanced signal detector, adaptive frequency domain equalizer, and a soft-decision Viterbi decoder help to alleviate multi-path effects and mutual interference in the reception of multiple streams.

The Wi-Fi unit supports fast receiver Automatic Gain Control (AGC) with synchronous and asynchronous control loops among antennas, and adaptive transmit power control function to obtain better performance in the analog portions of the transceiver.

19.1.2 Features

- General
 - CMOS MAC, Baseband PHY, and RF in a single chip for 802.11b/g/n compatible WLAN
 - Complete 802.11n solution for 2.4GHz band
 - 65Mbps receive PHY rate and 65Mbps transmit PHY rate using 20MHz bandwidth
 - Compatible with 802.11n specification
 - Backward compatible with 802.11b/g devices while operating in 802.11n mode
- Standards Supported
 - 802.11b/g/n compatible WLAN
 - 802.11e QoS Enhancement (WMM)
 - 802.11i (WPA, WPA2). Open, shared key, and pair-wise key authentication services

- Wi-Fi Direct support
- WLAN PHY Features
 - 802.11n OFDM
 - One Transmit and one Receive path (1T1R)
 - Support 2.4GHz band channels
 - 20MHz bandwidth transmission
 - DSSS with DBPSK and DQPSK, CCK modulation with long and short preamble
 - OFDM with BPSK, QPSK, 16QAM, and 64QAM modulation. Convolutional Coding Rate: 1/2, 2/3, 3/4, and 5/6
 - Maximum data rate 26Mbps in 802.11g and 65Mbps in 802.11n
 - Fast receiver Automatic Gain Control (AGC)
 - On-chip ADC and DAC
- Wi-Fi MAC Features
 - Frame aggregation for increased MAC efficiency (A-MSDU, A-MPDU)
 - Long NAV for media reservation with CF-End for NAV release
 - PHY-level spoofing to enhance legacy compatibility
 - Power saving mechanism
- Other Features
 - Supports the power save mode
 - Supports Wake-On-WLAN via Magic Packet and Wake-up frame
 - Support AES/TKIP group key update
 - Support Rate Adaptive function

19.1.3 Architecture

The Ameba-Z II Wi-Fi unit is combines a Wi-Fi MAC, a 1T1R capable Wi-Fi baseband, and RF in a single chip. A simplified block diagram of the Wi-Fi unit is illustrated in Figure 19-1.

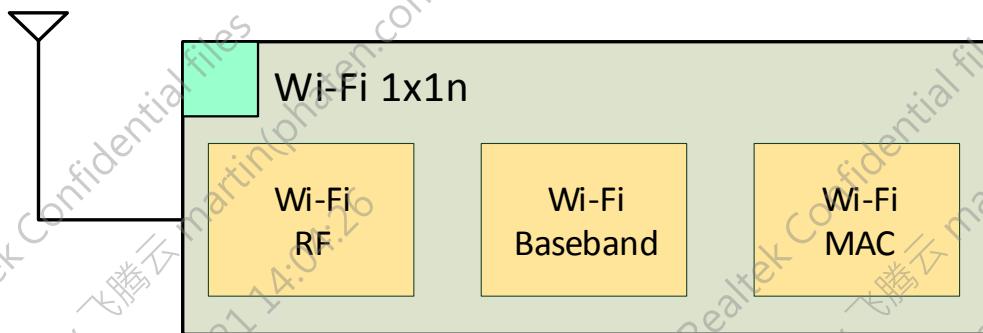


Figure 19-1 Wi-Fi Block Diagram

- **Wi-Fi MAC:** The main function is to control access the shared medium. The wireless

interface is a shared medium through which all stations and access point attempt to transmit data.

- Wi-Fi Baseband: It is used to process the down-converted digital signal to retrieve essential data for the wireless digital system.
- Wi-Fi RF: To process the transmission and reception in the target frequency band and channel.

19.2 Functional Description

19.2.1 Overview

The Realtek Wi-Fi units based on the IEEE 802.11 standard family and the OSI multi-layer protocol stack. They depend on the physical layer and uses radio waves to transmit and receive packets composed by the upper layers. The Wi-Fi baseband component allows the user data to be process in the digital domain between an end application and the transceiver device. The Wi-Fi MAC component provides flow control and multiplexing for the transmission medium. When user would like to sending data to another device on the wireless network, the Wi-Fi MAC component encapsulates higher-level frames into frames appropriate for the transmission medium and adds the frame check sequence to identify transmission errors, and then transfers the data to the physical layer as soon as the appropriate channel access method allows it.

19.2.2 Wi-Fi Unit Transmit/Receive Flow

19.2.2.1 Overview

The Wi-Fi unit included the Wi-Fi MAC、Wi-Fi baseband and Wi-Fi RF components. The mainly service of the data access was provide by the Wi-Fi MAC component. The Wi-Fi MAC component is responsible for coordinating access to the physical interface to the Access Point (AP) and station within the range can communicate effectively. The Wi-Fi MAC component takes data from a higher sub-layer called LLC, adds header and tail, and then sends them to physical layer for transmission. The reverse happens when receiving data from the physical layer. If the peer receive the frame in error, the Wi-Fi MAC component can retransmit.

19.2.2.2 Wi-Fi MAC TRx Flow

The data to be transmitted through the Wi-Fi interface will be prepared in the network driver layer and placed into a pre-allocated memory block, and then the MAC unit will be informed through the specified register. The TxDMA unit was used to data access if there is any data that needs to be readout from the specified memory and filled into the FIFO. After this, the MACTX unit will make some formats confirmation and decide whether to pass the data to the baseband and RF. The process of reception flow is reversed.

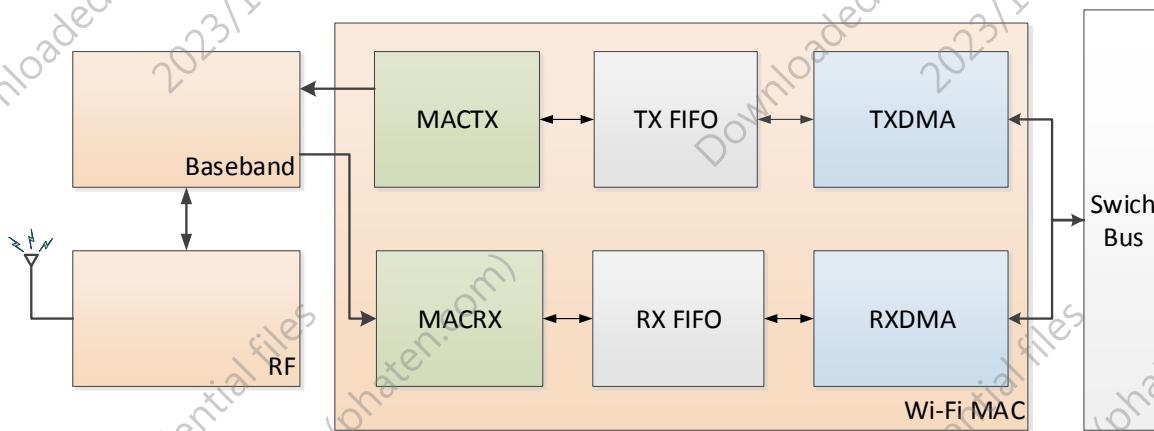


Figure 19-2 Wi-Fi MAC TRx Flow

- Switch Bus: The path for data exchange between the Wi-Fi unit and other peripheral unit.
- T/RXDMA: Transfer the packet data from the specified memory to the target location without CPU intervention.
- T/RX FIFO: Holds the transfer/received packet data.
- MACT/RX: The component is used to transfer/receive the packet data to physical interface and T/RX FIFO.

19.2.3 Wi-Fi Power Save Mode

19.2.3.1 Overview

The Power Save Mode is an important feature for the user to save the power of battery and extend the operating time of the device. It allows the devices to change the power state between active and sleep to saving the energy.

There are only two power management modes:

- Active mode
 - The station transmits or receives data at any time. In this mode, the RF circuit of station always turns on, and power consumption increases.

■ Power save mode

- The station can only transmit or receive data at a specific time. The RF circuit of station will turn off in the rest of the time. In this mode, station is able to reduce the power consumption and improve the battery life.

19.2.3.2 The State Machine of The Power Save Mode

The state machine can divide into two states based on the power modes of the STA:

- Active state: The STA is in active mode and is able to receive the packet at any time.
- Power save state: The STA is in a Power Save mode. AP shall buffer packet for the STA operating in a Power Save mode.

To change power mode of STA shall inform the AP using the power bit within the transmitted packet. The STA shall keep on its current state until the successful frame exchanges that includes an acknowledgment from the AP. A STA will transmit a packet that power bit set to 1 in order to register Power Save mode with the AP when the state machine change S0 to S1. Similarly, A STA will initiate a packet that power bit set to 0 in order to notify the AP is now in active mode when the state machine change S1 to S0.

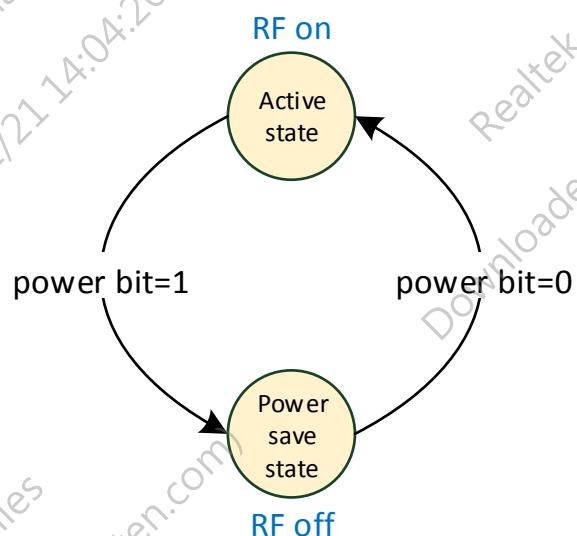


Figure 19-3 The State Machine of The Power Save Mode

19.2.3.3 The Behavior of The Power Save Mode

The Realtek power save mode can sleep for extended periods to avoid using the wireless network interface, the number of Beacon periods for which the station may choose to sleep, but the longer listen intervals require more buffer space on AP.

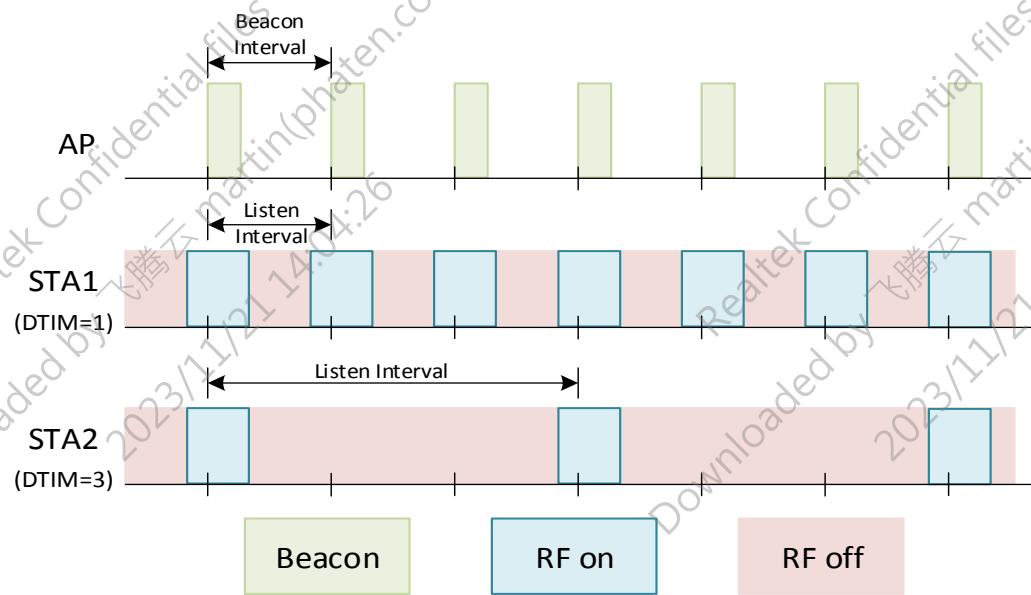


Figure 19-4 The Behavior of The Power Save Mode

19.2.4 Wi-Fi Wake On Wireless LAN

19.2.4.1 Overview

The Wake on Wireless LAN (WoWLAN) is a feature that allows remote wake-up of workstations from standby/sleep mode to facilitate device management.

19.2.4.2 The Behavior of The Wake On Wireless LAN

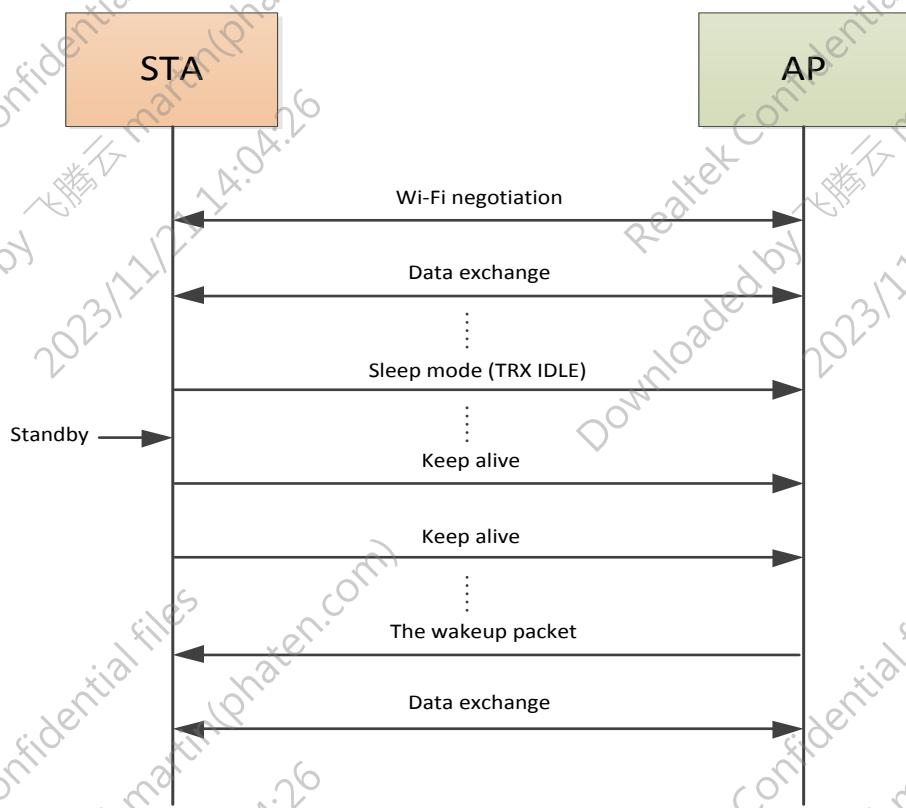


Figure 19-5 The Behavior of The Wake On Wireless LAN

There are two ways to wake up host for device management and two functions for Wi-Fi connection control.

■ Magic Packet Wakeup

- A Magic Packet frame must also meet the format of basic requirements.
- The destination address may be the node ID of the receiving station or a multicast address, including the broadcast address.

■ Unicast Packet Wakeup

- Wake up Host, when receiving all unicast packet.

■ Disconnection decision

- Received the Disassociation or Deauthentication packet.
- The device didn't receive the reply from the AP for a period of time.

■ Keep Alive

- Under sleep mode, station needs periodically to inform associated AP to keep alive. This can keep the internet connection capability during sleep.

20 Bluetooth

20.1 Overview

The Ameba-Z II highly integrated Bluetooth Low Energy controller with a UART interface. It combines a BLE Protocol (PHY, LL, L2CAP, SM, ATT, GAP, GATT), BLE Baseband, Modem, and BLE RF in chip, also supports BLE user GATT-based profile application. Profile is one of the modules constituting SDK, which packages underlying implementation details for Low Energy protocol stack, and provides user-friendly and easy-to-use interfaces for use in development of application.

20.1.1 Features

Table 20-1 Supported BT Features

Specification Version	BT Feature	Note
BT 4.0	Advertiser	
	Scanner	
	Initiator	
	Master	Maximum number is 1
	Slave	Maximum number is 1
BT 4.1	LE Scatternet	1 Master + 1 Slave
BT 4.2	LE Secure Connections	

20.1.2 BLE Profile Architecture

Definition of profile in Bluetooth specification is different from that of Protocol. Protocol is defined as layer protocols in Bluetooth specification such as Link Layer, Logical Link Control and Adaptation protocol (L2CAP), Security Manager protocol (SMP), and Attribute protocol (ATT), while Profile involves implementation of interoperability of Bluetooth applications from the perspective of how to use layer protocols in Bluetooth specification. Profile defines features and functions that are available in Protocol, and implementation of interaction details between devices, so as to accommodate Bluetooth protocol stack to application development in various scenarios.

The relationship between Profile and Protocol in Bluetooth specification is shown in Figure 20-1.

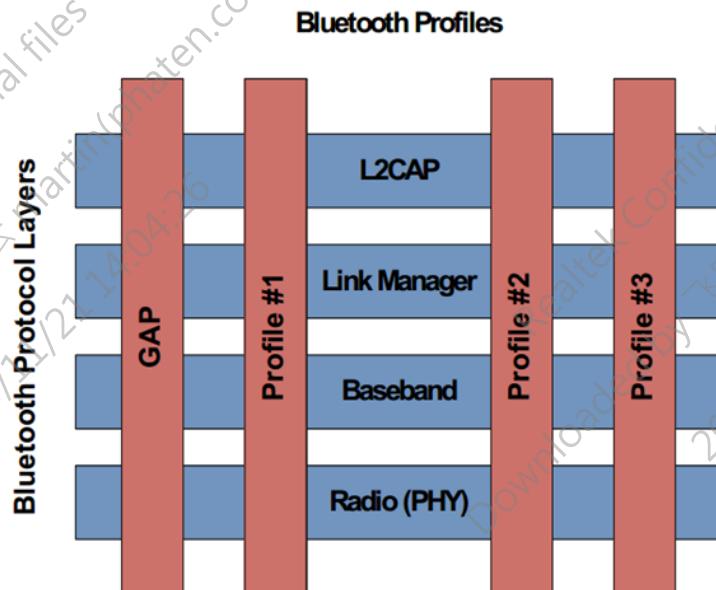


Figure 20-1 Bluetooth Profiles

As shown in Figure 20-1, Profile is illustrated in red rectangular, GAP, Profile #1, Profile #2, and Profile #3. Profiles in Bluetooth specification are classified into two types - GAP and GATT Based Profile (Profile #1, Profile #2 and Profile #3).

20.1.2.1 GAP

GAP is basic Profile which must be implemented by all Bluetooth devices, and used to describe actions and methods including device discovery, connection, security requirement, and authentication. GAP for Bluetooth Low Energy also defines 4 application roles - Broadcaster, Observer, Peripheral and Central - for optimization in various application scenarios.

Broadcaster is applicable to applications sending data only via broadcast. Observer is applicable to applications receiving data via broadcast. Peripheral is applicable to applications setting link connection. Central is applicable to applications setting a single or multiple link connections

20.1.2.2 GATT Based Profile

In Bluetooth specification, another commonly used Profile is GATT Based Profile. GATT is a standard based on server-client interaction defined in Bluetooth specification, and is used to implement provision of service data and access to service data. GATT Based Profile is a standard which is defined based on server-client interaction to meet various application cases and used for data interaction between devices as specified. Profile is made up in the form of

Service and Characteristic, as shown in Figure 20-2.

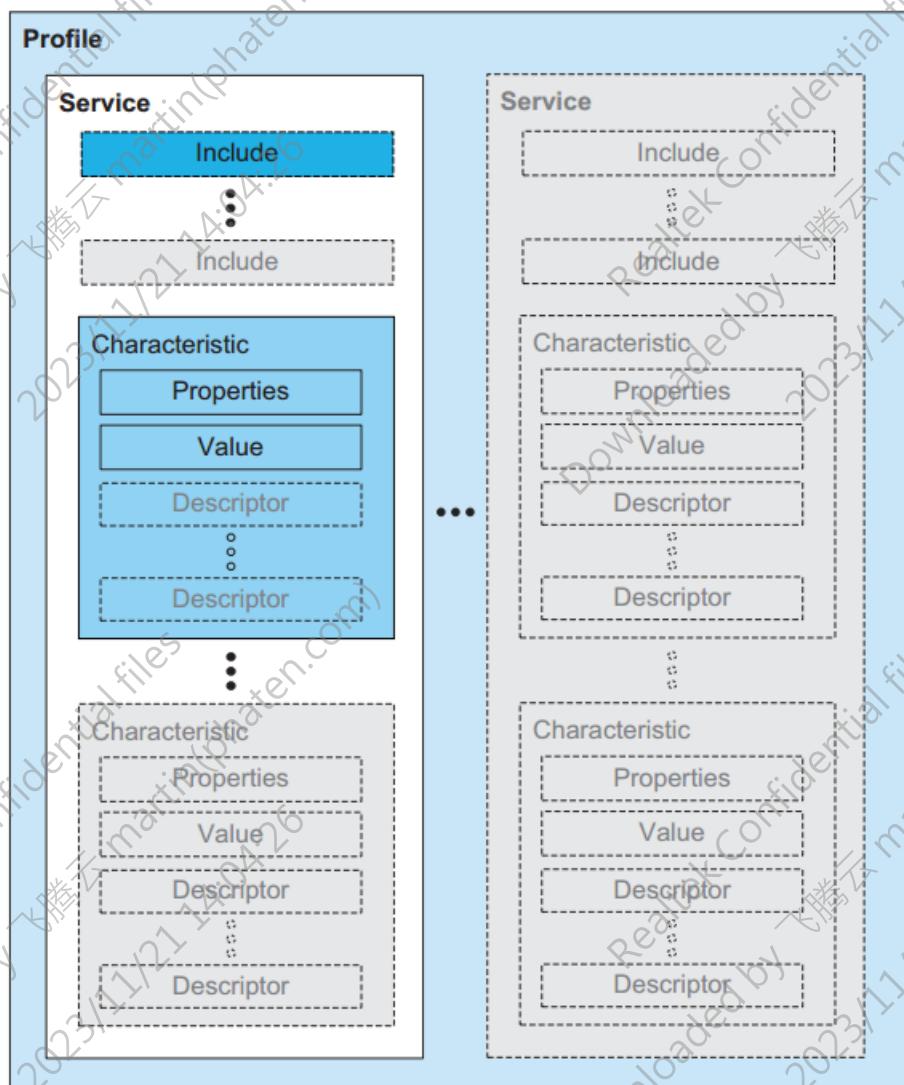


Figure 20-2 GATT Based Profile Hierarchy

Appendix A: Abbreviations

A

AES	Advanced Encryption Standard
AGC	Automatic Gain Control
AP	Access Point

B

BB	Baseband
BCN	Beacon
BLE	Bluetooth Low Energy
BT	Bluetooth

C

CBC	Cipher Block Chaining
CFB	Cipher Feedback
CRC	Cyclic Redundancy Check
CTR	Counter (Cipher)

D

DMAC	Direct Memory Access Controller
DTCM	Data TCM

E

ECB	Electronic Codebook
-----	---------------------

F

FIFO	First In First Out
------	--------------------

G

GPIO	General Purpose Input/Output
------	------------------------------

H

HMAC	Hash-Based Message Authentication Code
------	--

I

I ² C	Inter-Integrated Circuit
------------------	--------------------------

IOT	Internet of Things
ITCM	Instruction TCM
J	
JTAG	Joint Test Action Group

K	
L	
LDO	Low Dropout Regulator
LP	Low Power
LPC	Low Pin Count

M	
MAC	Media Access Control
MCU	Micro Control Unit
MPDU	MAC Protocol Data Unit
MPU	Memory Protection Unit
MSDU	MAC Service Data Unit

N	
NVIC	Nested Vectored Interrupt Controller

O	
OTP	One Time Programmable
OFDM	Orthogonal Frequency-Division Multiplexing

P	
PMU	Power Management Unit
pSRAM	Pseudo Static Random Access Memory
PWM	Pulse Width Modulation

Q	
QPI	Quad Peripheral Interface

R	
ROM	Read Only Memory

S

SCE	Security Code Engine
SDIO	Secure Digital Input/Output
SoC	System on a chip
SPI	Serial Peripheral Interface
SWD	Serial Wire Debug
SWR	Switch Regulator

T

TCM	Tightly Coupled Memory
TKIP	Temporal Key Integrity Protocol

U

UART	Universal Asynchronous Receiver/Transmitter
------	---

V**W**

WDT	Watch Dog Timer
WMM	Wi-Fi Multimedia
WPA	Wi-Fi Protected Access

X

XIP	Execution In Place
-----	--------------------

Y**Z**

Appendix B: ARM ACK Certificate



16 August 2018

CONFIDENTIAL

I ES-LTM-25162
SP-Version: 1.0

ARM Ltd
110 Fulbourn Road
Cambridge
United Kingdom
CB1 9NJ

T+44 1223 400400
www.arm.com

To whom it may concern,

Arm certifies that as of 1st August 2018, the following cores produced by Realtek Semiconductor Corporation whose principal place of business is situated at No.2, Innovation Road II, Hsinchu Science Park, 300 Hsinchu, Taiwan, 2 ("Realtek") have been verified as passing the Arm architecture compliance kit ("ACK") for the stated version of the Arm Architecture:

Core	Architecture	ACK Version
Real-M300 (KM4)	Armv8.0-M Mainline	AR100-VA-01008-r0p0-03eac0

Realtek is responsible for the quality and correctness of their Arm architecture compliant core designs and only use the ACK as part of the process of demonstrating compliance to the Arm Architecture.

Details of the Arm Architecture can be found on Arm's website: www.arm.com.

Sincerely,



Richard Grisenthwaite
Senior Vice President, Chief Architect and Fellow

Figure B-1 ARM ACK Certificate