



BL616/BL618

Reference Manual

Version: 0.91

Copyright @ 2022

Contents

1	System and Memory	26
1.1	System Architecture	26
1.2	CPU	27
1.2.1	Features	27
1.2.2	Extended Functions	28
1.2.3	Implemented Standards	28
1.3	Boot	29
1.4	Address Mapping	29
1.5	Interrupt Source	30
1.6	Peripheral Overview	32
2	Reset and Clock	34
2.1	Overview	34
2.2	Reset Management	34
2.3	Clock Source	35
3	GLB	39
3.1	Overview	39
3.2	Functional Description	39
3.2.1	Clock Management	39
3.2.2	Reset Management	39
3.2.3	Bus Management	40
3.2.4	Memory Management	40
3.2.5	LDO Management	40
4	GPIO	41
4.1	Overview	41
4.2	Features	41
4.3	GPIO function list	41

4.4	GPIO Input Settings	43
4.5	GPIO Output Settings	43
4.5.1	Normal Output Mode	43
4.5.2	Set/Clear Output Mode	43
4.6	I/O FIFO	44
4.7	I/O Interrupt	44
4.8	Register description	45
4.8.1	gpio_cfg0	47
4.8.2	gpio_cfg1	49
4.8.3	gpio_cfg2	50
4.8.4	gpio_cfg3	52
4.8.5	gpio_cfg4	54
4.8.6	gpio_cfg5	56
4.8.7	gpio_cfg6	57
4.8.8	gpio_cfg7	59
4.8.9	gpio_cfg8	61
4.8.10	gpio_cfg9	63
4.8.11	gpio_cfg10	64
4.8.12	gpio_cfg11	66
4.8.13	gpio_cfg12	68
4.8.14	gpio_cfg13	70
4.8.15	gpio_cfg14	72
4.8.16	gpio_cfg15	74
4.8.17	gpio_cfg16	76
4.8.18	gpio_cfg17	78
4.8.19	gpio_cfg18	80
4.8.20	gpio_cfg19	82
4.8.21	gpio_cfg20	84
4.8.22	gpio_cfg21	86
4.8.23	gpio_cfg22	88
4.8.24	gpio_cfg23	90
4.8.25	gpio_cfg24	92
4.8.26	gpio_cfg25	94
4.8.27	gpio_cfg26	96
4.8.28	gpio_cfg27	98
4.8.29	gpio_cfg28	100
4.8.30	gpio_cfg29	102
4.8.31	gpio_cfg30	104
4.8.32	gpio_cfg31	106

4.8.33	gpio_cfg32	108
4.8.34	gpio_cfg33	110
4.8.35	gpio_cfg34	112
4.8.36	gpio_cfg128	114
4.8.37	gpio_cfg129	116
4.8.38	gpio_cfg136	116
4.8.39	gpio_cfg137	118
4.8.40	gpio_cfg138	118
4.8.41	gpio_cfg139	121
4.8.42	gpio_cfg141	125
4.8.43	gpio_cfg142	125
4.8.44	gpio_cfg143	126
4.8.45	gpio_cfg144	127
5	ADC	129
5.1	Overview	129
5.2	Features	129
5.3	Functional Description	130
5.3.1	ADC Pins and Internal Signals	131
5.3.2	ADC Channels	131
5.3.3	ADC Clocks	132
5.3.4	ADC Conversion Mode	133
5.3.5	ADC Result	133
5.3.6	ADC Interrupt	134
5.3.7	ADC FIFO	135
5.3.8	ADC Setup Process	135
5.3.9	VBAT Measurement	136
5.3.10	TSEN Measurement	136
5.4	Register description	137
5.4.1	gpadc_config	137
5.4.2	gpadc_dma_rdata	138
6	DAC	139
6.1	Overview	139
6.2	Features	139
6.3	Functional Description	139
6.3.1	DAC channel enable	140
6.3.2	DAC data format	141
6.3.3	DAC output voltage	141
6.3.4	DAC conversion	142
6.3.4.1	CPU mode	142

6.3.4.2	DMA method	142
6.4	Register description	143
6.4.1	gpdac_config	143
6.4.2	gpdac_dma_config	144
6.4.3	gpdac_dma_wdata	145
7	DMA	146
7.1	Overview	146
7.2	Features	146
7.3	Functional Description	147
7.3.1	Operating Principle	147
7.3.2	DMA Channel Configuration	149
7.3.3	Supported Peripherals	149
7.3.4	Linked List Mode	151
7.3.5	DMA Interrupt	152
7.4	Transfer Mode	153
7.4.1	Memory to Memory	153
7.4.2	Memory to Peripherals	153
7.4.3	Peripheral to Memory	154
7.5	Register description	154
7.5.1	DMA_IntStatus	156
7.5.2	DMA_IntTCStatus	156
7.5.3	DMA_IntTCClear	156
7.5.4	DMA_IntErrorStatus	157
7.5.5	DMA_IntErrClr	157
7.5.6	DMA_RawIntTCStatus	158
7.5.7	DMA_RawIntErrorStatus	158
7.5.8	DMA_EnbldChns	159
7.5.9	DMA_SoftBReq	159
7.5.10	DMA_SoftSReq	159
7.5.11	DMA_SoftLBReq	160
7.5.12	DMA_SoftLSReq	160
7.5.13	DMA_Config	160
7.5.14	DMA_Sync	161
7.5.15	DMA_C0SrcAddr	161
7.5.16	DMA_C0DstAddr	162
7.5.17	DMA_C0LLI	162
7.5.18	DMA_C0Control	162
7.5.19	DMA_C0Config	164
7.5.20	DMA_C0RSVD	165

7.5.21	DMA_C1SrcAddr	165
7.5.22	DMA_C1DstAddr	165
7.5.23	DMA_C1LLI	166
7.5.24	DMA_C1Control	166
7.5.25	DMA_C1Config	168
7.5.26	DMA_C1RSVD	168
7.5.27	DMA_C2SrcAddr	169
7.5.28	DMA_C2DstAddr	169
7.5.29	DMA_C2LLI	170
7.5.30	DMA_C2Control	170
7.5.31	DMA_C2Config	171
7.5.32	DMA_C2RSVD	172
7.5.33	DMA_C3SrcAddr	173
7.5.34	DMA_C3DstAddr	173
7.5.35	DMA_C3LLI	173
7.5.36	DMA_C3Control	174
7.5.37	DMA_C3Config	175
7.5.38	DMA_C3RSVD	176
8	IR	177
8.1	Overview	177
8.2	Features	177
8.3	Functional Description	177
8.3.1	Fixed Protocol Based Receiving	177
8.3.2	Pulse width receiving	179
8.3.3	IR Interrupt	179
8.4	Register description	179
8.4.1	irrx_config	180
8.4.2	irrx_int_sts	181
8.4.3	irrx_pw_config	182
8.4.4	irrx_data_count	182
8.4.5	irrx_data_word0	182
8.4.6	irrx_data_word1	183
8.4.7	ir_fifo_config_0	183
8.4.8	ir_fifo_config_1	184
8.4.9	ir_fifo_rdata	184
9	SPI	185
9.1	Overview	185
9.2	Features	185

9.3	Functional Description	186
9.3.1	Clock Control	186
9.3.2	Master Continuous Transfer Mode	187
9.3.3	Master-Slave: Transfer and Receive Data	187
9.3.4	Receive Ignore Function	187
9.3.5	Filtering Function	188
9.3.6	Configurable MSB/LSB Transfer	188
9.3.7	Adjustable Byte Transfer Sequence	189
9.3.8	Slave Mode Timeout Mechanism	189
9.3.9	I/O Transfer Mode	189
9.3.10	DMA Transfer Mode	189
9.3.11	SPI Interrupt	190
9.4	Register description	190
9.4.1	spi_config	191
9.4.2	spi_int_sts	192
9.4.3	spi_bus_busy	194
9.4.4	spi_prd_0	194
9.4.5	spi_prd_1	195
9.4.6	spi_rxd_ignr	195
9.4.7	spi_sto_value	196
9.4.8	spi_fifo_config_0	196
9.4.9	spi_fifo_config_1	197
9.4.10	spi_fifo_wdata	197
9.4.11	spi_fifo_rdata	198
9.4.12	backup_io_en	199
10	UART	200
10.1	Overview	200
10.2	Features	200
10.3	Functional Description	201
10.3.1	Data Formats	201
10.3.2	Basic Architecture	202
10.3.3	Transmitter	202
10.3.4	Receiver	203
10.3.5	Baud Rate Setting	203
10.3.6	Filtering	204
10.3.7	Automatic Baud Rate Detection	204
10.3.8	Hardware Flow Control	205
10.3.9	DMA Transfer	206
10.3.10	Support for LIN Bus	206

10.3.11 RS485 mode	208
10.3.12 UART Interrupt	208
10.3.13 TX/RX end of transfer interrupt	209
10.3.14 TX/RX FIFO request interrupt	210
10.3.15 RX timeout interrupt	210
10.3.16 RX parity check error interrupt	210
10.3.17 TX/RX FIFO overflow interrupt	211
10.3.18 RX BCR interrupt	211
10.3.19 LIN synchronization error interrupt	211
10.3.20 Auto baud rate detection (universal/fixed characters mode) interrupt	211
10.4 Register description	211
10.4.1 utx_config	212
10.4.2 urx_config	214
10.4.3 uart_bit_prd	215
10.4.4 data_config	215
10.4.5 utx_ir_position	216
10.4.6 urx_ir_position	216
10.4.7 urx_rto_timer	216
10.4.8 uart_sw_mode	217
10.4.9 uart_int_sts	218
10.4.10 uart_int_mask	219
10.4.11 uart_int_clear	220
10.4.12 uart_int_en	221
10.4.13 uart_status	222
10.4.14 sts_urx_abr_prd	222
10.4.15 urx_abr_prd_b01	223
10.4.16 urx_abr_prd_b23	223
10.4.17 urx_abr_prd_b45	223
10.4.18 urx_abr_prd_b67	224
10.4.19 urx_abr_pw_tol	224
10.4.20 urx_bcr_int_cfg	225
10.4.21 utx_rs485_cfg	225
10.4.22 uart_fifo_config_0	226
10.4.23 uart_fifo_config_1	226
10.4.24 uart_fifo_wdata	227
10.4.25 uart_fifo_rdata	228
11 I2C	229
11.1 Overview	229
11.2 Features	229

11.3	Functional Description	230
11.3.1	Start and stop conditions	230
11.3.2	Data Transfer Format	230
11.3.3	Arbitration	233
11.4	I2C Clock Setting	233
11.5	I2C Configuration Flow	234
11.5.1	Configuration Items	234
11.5.2	Read/Write Flag Bit	234
11.5.3	Slave Address	234
11.5.4	Slave Register Address	234
11.5.5	Slave Register Address Length	234
11.5.6	Data	235
11.5.7	Data Length	235
11.5.8	Enable Signal	235
11.6	FIFO Management	236
11.7	Use with DMA	236
11.7.1	DMA Sending Flow	236
11.7.2	DMA Receiving Flow	237
11.8	Interrupt	237
11.9	Register description	238
11.9.1	i2c_config	238
11.9.2	i2c_int_sts	239
11.9.3	i2c_sub_addr	241
11.9.4	i2c_bus_busy	241
11.9.5	i2c_prd_start	242
11.9.6	i2c_prd_stop	242
11.9.7	i2c_prd_data	242
11.9.8	i2c_fifo_config_0	243
11.9.9	i2c_fifo_config_1	244
11.9.10	i2c_fifo_wdata	244
11.9.11	i2c_fifo_rdata	245
12	PWM	246
12.1	Overview	246
12.2	Features	246
12.3	Functional Description	247
12.3.1	Clock and Frequency Divider	247
12.3.2	Active Level	247
12.3.3	Principle of Pulse Generation	247
12.3.4	Brake	248

12.3.5	Dead Zone	249
12.3.6	Cycle and Duty Ratio Calculation	249
12.3.7	PWM Interrupt	249
12.3.8	ADC Linkage	249
12.4	Register description	250
12.4.1	pwm_int_config	250
12.4.2	pwm_mc0_config0	251
12.4.3	pwm_mc0_config1	252
12.4.4	pwm_mc0_period	254
12.4.5	pwm_mc0_dead_time	254
12.4.6	pwm_mc0_ch0_thre	255
12.4.7	pwm_mc0_ch1_thre	256
12.4.8	pwm_mc0_ch2_thre	256
12.4.9	pwm_mc0_ch3_thre	257
12.4.10	pwm_mc0_int_sts	257
12.4.11	pwm_mc0_int_mask	258
12.4.12	pwm_mc0_int_clear	259
12.4.13	pwm_mc0_int_en	260
13	TIMER	261
13.1	Overview	261
13.2	Features	262
13.3	Functional Description	262
13.3.1	Working Principle of General Purpose Timer	263
13.3.2	Working Principle of Watchdog Timer	265
13.3.3	Alarm Setting	265
13.3.4	Watchdog Alarm	266
13.4	Register description	266
13.4.1	TCCR	268
13.4.2	TMR2_0	268
13.4.3	TMR2_1	269
13.4.4	TMR2_2	269
13.4.5	TMR3_0	269
13.4.6	TMR3_1	270
13.4.7	TMR3_2	270
13.4.8	TCR2	270
13.4.9	TCR3	271
13.4.10	TSR2	271
13.4.11	TSR3	272
13.4.12	TIER2	272

13.4.13	TIER3	273
13.4.14	TPLVR2	273
13.4.15	TPLVR3	273
13.4.16	TPLCR2	274
13.4.17	TPLCR3	274
13.4.18	WMER	275
13.4.19	WMR	275
13.4.20	WVR	276
13.4.21	WSR	276
13.4.22	TICR2	277
13.4.23	TICR3	277
13.4.24	WICR	278
13.4.25	TCER	278
13.4.26	TCMR	279
13.4.27	TILR2	279
13.4.28	TILR3	280
13.4.29	WCR	280
13.4.30	WFAR	281
13.4.31	WSAR	281
13.4.32	TCVWR2	281
13.4.33	TCVWR3	282
13.4.34	TCVSYN2	282
13.4.35	TCVSYN3	282
13.4.36	TCDR	283
13.4.37	GPIO	283
13.4.38	GPIO_LAT1	284
13.4.39	GPIO_LAT2	284
14	I2S	285
14.1	Overview	285
14.2	Features	285
14.3	Functional Description	286
14.4	Functional Description	286
14.4.1	Data Formats	286
14.4.2	Basic architecture	289
14.4.3	Clock source	289
14.4.4	I2S Interrupt	289
14.5	Register description	290
14.5.1	i2s_config	290
14.5.2	i2s_int_sts	292

14.5.3	i2s_bclk_config	293
14.5.4	i2s_fifo_config_0	293
14.5.5	i2s_fifo_config_1	294
14.5.6	i2s_fifo_wdata	295
14.5.7	i2s_fifo_rdata	295
14.5.8	i2s_io_config	296
15	AudioPWM	297
15.1	Overview	297
15.2	Features	297
15.3	Clock Tree	297
15.4	Functional Description	298
15.4.1	AudioPWM Interrupt	298
15.4.2	FIFO Format Control	299
15.4.3	Startup of FIFO and DMA Transfer	300
15.4.4	Audio Channel Selector	300
15.4.5	Volume Control	302
15.4.6	ZeroDetect	302
15.5	Configuration Process	302
15.6	Register description	302
15.6.1	aupwm_0	303
15.6.2	aupwm_status	304
15.6.3	aupwm_s0	305
15.6.4	aupwm_s0_misc	307
15.6.5	aupwm_zd_0	308
15.6.6	aupwm_1	308
15.6.7	aupwm_rsvd	309
15.6.8	aupwm_test_0	309
15.6.9	aupwm_test_1	310
15.6.10	aupwm_test_2	310
15.6.11	aupwm_test_3	311
15.6.12	aupwm_fifo_ctrl	311
15.6.13	aupwm_fifo_status	313
15.6.14	aupwm_fifo_data	314
16	AudioADC	315
16.1	Overview	315
16.2	Features	315
16.3	Clock Tree	315
16.4	Functional Description	316
16.4.1	Selection of PDM Left and Right Channels	317

16.4.2	AUADC Interrupt	317
16.4.3	FIFO Format Control	317
16.4.4	Startup of FIFO and DMA Transfer	318
16.5	Configuration Process	318
16.6	Register description	319
16.6.1	audpdm_top	319
16.6.2	audpdm_if	320
16.6.3	pdm_adc_0	321
16.6.4	pdm_adc_1	321
16.6.5	pdm_dac_0	322
16.6.6	pdm_pdm_0	322
16.6.7	pdm_dbg_0	323
16.6.8	pdm_dbg_1	323
16.6.9	pdm_dbg_2	324
16.6.10	pdm_adc_s0	324
16.6.11	audadc_ana_cfg1	325
16.6.12	audadc_ana_cfg2	326
16.6.13	audadc_cmd	328
16.6.14	audadc_data	330
16.6.15	audadc_rx_fifo_ctrl	331
16.6.16	audadc_rx_fifo_status	334
16.6.17	audadc_rx_fifo_data	335
17	Emac	336
17.1	Overview	336
17.2	Features	336
17.3	Functional Description	337
17.4	Clock	339
17.5	RX/TX BD	339
17.6	PHY Interaction	340
17.7	Programming Flow	341
17.7.1	PHY initialization	341
17.7.2	Send Data Frame	342
17.7.3	Receive Data Frame	342
18	USB	344
18.1	Overview	344
19	ISO11898	345
19.1	Overview	345
19.2	Features	345

19.3 Functional Description	345
19.3.1 TX Buffer (TXB)	345
19.3.2 RX Buffer (RXB, RXFIFO)	346
19.3.3 Access Control Filter (ACF)	346
19.3.4 Bit Stream Processor (BSP)	346
19.3.5 Bit Timing Logic (BTL)	346
19.3.6 Error Management Logic (EML)	346
19.4 Functional Description	346
19.4.1 Mode	346
19.4.1.1 Self-test Mode	346
19.4.1.2 Silent Mode	347
19.4.1.3 Reset Mode	347
19.4.2 Sending Process	348
19.4.2.1 Process	348
19.4.2.2 Termination of Sending	349
19.4.2.3 Self-sending and Self-receiving	349
19.4.2.4 Precautions	349
19.4.3 Receiving Process	349
19.4.3.1 Process	349
19.4.3.2 Number of Messages	350
19.4.3.3 RXB	350
19.4.4 Identifier Filtering	350
19.4.4.1 Single Filter Configuration	351
19.4.4.2 Double Filter Configuration	352
19.4.5 Error Management	355
19.4.5.1 Arbitration Lost	355
19.4.5.2 Error Capture	357
19.4.5.3 RX Error Counter Register (RXERR)	358
19.4.5.4 TX Error Counter Register (TXERR)	358
19.4.5.5 Error Limit Setting	359
19.4.6 Bit Timing	359
19.4.6.1 Baud Rate Prescaler (BRP)	359
19.4.6.2 Synchronization Jump Width (SJW)	360
19.4.6.3 Sampling (SAM)	360
19.4.6.4 Time Segment (TSEG)	360
20 CAM	361
20.1 Overview	361
20.2 Features	361

20.3 Function description	362
20.3.1 DVP (Digital Video Port) signal and configuration	362
20.3.2 YCbCr format	362
20.3.3 Movie Mode/Photo Mode	363
20.3.4 RGB888 to RGB565/RGBA8888 output	363
20.3.5 Image rectangle crop	363
20.3.6 Frame rounding function	364
20.3.7 Line Frame Sync Signal Integrity Detection	364
20.3.8 Cache image information	364
20.3.9 Support a variety of interrupt information (can be configured independently of the switch)	365
20.4 Register description	365
20.4.1 dvp2axi_configue	366
20.4.2 dvp2axi_addr_start	368
20.4.3 dvp2axi_mem_bcnt	369
20.4.4 dvp_status_and_error	369
20.4.5 dvp2axi_frame_bcnt	370
20.4.6 dvp_frame_fifo_pop	371
20.4.7 dvp2axi_frame_vld	371
20.4.8 dvp2axi_frame_period	372
20.4.9 dvp2axi_misc	372
20.4.10 dvp2axi_hsync_crop	373
20.4.11 dvp2axi_vsync_crop	373
20.4.12 dvp2axi_fram_exm	374
20.4.13 frame_start_addr0	374
20.4.14 frame_start_addr1	374
20.4.15 frame_start_addr2	375
20.4.16 frame_start_addr3	375
20.4.17 frame_id_sts01	375
20.4.18 frame_id_sts23	376
20.4.19 dvp_debug	376
20.4.20 dvp_dummy_reg	377
21 MJPEG	378
21.1 Overview	378
21.2 Features	378
21.3 Function description	379
21.3.1 Input configuration	379
21.3.2 Quantization coefficient table	379
21.3.3 Software mode and link mode	379

21.3.4	Swap mode	380
21.3.5	Kick mode	380
21.3.6	Jpg function	380
21.3.7	Cache image information	380
21.3.8	Support a variety of interrupt information (can be configured independently of the switch)	380
22	DBI	382
22.1	Overview	382
22.2	Features	382
22.3	Function description	383
22.3.1	DBI Type B	383
22.3.1.1	Write timing	383
22.3.1.2	Read timing	385
22.3.1.3	Output RGB565	385
22.3.1.4	Output RGB666	386
22.3.1.5	Output RGB888	387
22.3.2	DBI Type C 3-Line	388
22.3.2.1	Write timing	388
22.3.2.2	Read timing	389
22.3.2.3	Output RGB565	390
22.3.2.4	Output RGB666	390
22.3.2.5	Output RGB888	391
22.3.3	DBI Type C 4-Line	391
22.3.3.1	Write timing	391
22.3.3.2	Read timing	392
22.3.3.3	Output RGB565	392
22.3.3.4	Output RGB666	393
22.3.3.5	Output RGB888	394
22.3.4	QSPI	394
22.3.4.1	Write timing	394
22.3.4.2	Read timing	395
22.3.4.3	1-Wire Command, 1-Wire Address and 1-Wire Data Pattern	396
22.3.4.4	1-Wire Command, 1-Wire Address and 4-Wire Data Pattern	398
22.3.4.5	1-Wire Command, 4-Wire Address and 4-Wire Data Pattern	401
22.3.5	Input pixel format	402
22.3.6	Configuration of CS Signal Pull-up Release Condition	404
22.3.7	Interrupt	404
22.3.8	DMA	404

23	SDH	405
23.1	Overview	405
23.2	Features	405
23.3	Functional Description	405
23.3.1	SDH Overall Structure	406
23.3.2	Register Mapping	407
23.3.3	Support for Multiple Card Slots	408
23.3.4	Supporting DMA	409
23.3.5	SD Command Generation	409
23.3.6	Suspend and Resume Mechanism	410
23.3.7	Buffer Control	411
23.3.7.1	Control of Buffer Pointer	411
23.3.7.2	Determination of Buffer Block Length	411
23.3.7.3	Dividing Large Data Transfer	411
23.3.8	Relationship Between Interrupt Control Registers	411
23.3.9	Hardware Block Diagram and Timing Part	412
23.3.10	Auto CMD12	413
23.3.11	Controlling SDCLK	413
23.3.12	Advanced DMA	414
23.3.12.1	Block Diagram of ADMA2	415
23.3.12.2	Data Address and Data Length Requirements	416
23.3.12.3	Descriptor Table	417
23.3.12.4	ADMA2 State	418
23.3.13	Test Register	419
23.3.14	Block Count	419
23.3.15	Sampling Clock Tuning	420
23.3.16	SD Host Standard Register	420
23.3.16.1	SD Host Control Register Mapping	420
23.3.16.2	Configuration of Register Type	422
23.3.16.3	Initial Value of Register	423
23.3.16.4	Reserved Bits of a Register	423
24	SDIO	424
24.1	Overview	424
24.2	Features	424
24.3	Functional Description	424
24.3.1	Definition of SDIO Signal	424
24.3.1.1	SDIO Card Type	424
24.3.1.2	SDIO Card Mode	425
24.3.1.3	Signal Pin	425

24.3.2 SDIO Card Initialization	425
24.3.2.1 Difference in I/O Card Initialization	426
24.3.2.2 IO_SEND_OP_COND command (CMD5)	428
24.3.2.3 IO_SEND_OP_COND Response (R4)	428
24.3.2.4 Special Initialization Considerations for combo Cards	430
24.3.3 Differences with SD Memory Specification	430
24.3.3.1 SDIO Command List	430
24.3.3.2 Unsupported SD Memory Commands	433
24.3.3.3 Modified R6 Response	434
24.3.3.4 Reset for SDIO	434
24.3.3.5 Bus Width	435
24.3.3.6 Card Detect Resistor	435
24.3.3.7 Data Transfer Block Sizes	435
24.3.3.8 Data Transfer Abort	436
24.3.4 New I/O Read/Write Commands	436
24.3.4.1 IO_RW_DIRECT Command (CMD52)	436
24.3.4.2 IO_RW_DIRECT Response (R5)	437
24.3.4.3 IO_RW_EXTENDED Command (CMD53)	438
24.3.5 SDIO Card Internal Operation	439
24.3.5.1 Overview	440
24.3.5.2 Register Access Time	441
24.3.5.3 Interrupt	442
24.3.5.4 Suspend/Resume	442
24.3.5.5 Read Wait	442
24.3.5.6 CMD52 During Data Transfer	443
24.3.5.7 Fixed Internal Mapping	443
24.3.5.8 Common I/O Area (CIA)	443
24.3.5.9 Card Common Control Registers (CCCR)	443
24.3.5.10 Function Basic Registers (FBR)	444
24.3.5.11 Card Information Structure (CIS)	444
24.3.5.12 Multiple Function SDIO Cards	444
24.3.5.13 Setting Block Size with CMD53	445
24.3.5.14 Bus State Diagram	445
24.3.6 Embedded I/O Code Storage Area (CSA)	447
24.3.6.1 CSA Access	447
24.3.6.2 CSA Data Format	447
24.3.7 SDIO Interrupts	447
24.3.7.1 SPI and SD 1-Bit Mode Interrupts	448
24.3.7.2 SD 4-Bit Mode Interrupt	448

24.3.7.3	Interrupt Clear Timing	448
24.3.8	SDIO Suspend/Resume Operation	448
24.3.9	SDIO Read Wait Operation	449
25	LowPower	450
25.1	Overview	450
25.2	Features	451
25.3	Functional Description	451
25.3.1	Power Domain	451
25.3.2	Wake-up Source	453
25.3.3	Power Modes	454
25.3.4	IO Retention	455
25.4	Register description	456
25.4.1	PDS_CTL	456
25.4.2	PDS_TIME1	459
25.4.3	PDS_INT	459
25.4.4	PDS_CTL2	460
25.4.5	PDS_CTL3	461
25.4.6	PDS_CTL4	462
25.4.7	pds_stat	464
25.4.8	pds_ram1	465
25.4.9	PDS_CTL5	467
25.4.10	PDS_RAM2	468
25.4.11	pds_gpio_i_set	469
25.4.12	pds_gpio_pd_set	469
25.4.13	pds_gpio_int	471
25.4.14	pds_gpio_stat	473
25.4.15	PDS_RAM3	473
25.4.16	PDS_RAM4	474
25.5	Register description	475
25.5.1	HBN_CTL	476
25.5.2	HBN_TIME_L	477
25.5.3	HBN_TIME_H	478
25.5.4	RTC_TIME_L	478
25.5.5	RTC_TIME_H	478
25.5.6	HBN_IRQ_MODE	479
25.5.7	HBN_IRQ_STAT	480
25.5.8	HBN_IRQ_CLR	481
25.5.9	HBN_PIR_CFG	481
25.5.10	HBN_PIR_VTH	482

25.5.11	HBN_PIR_INTERVAL	482
25.5.12	HBN_BOR_CFG	483
25.5.13	HBN_GLB	484
25.5.14	HBN_SRAM	485
25.5.15	HBN_PAD_CTRL_0	486
25.5.16	HBN_PAD_CTRL_1	487
25.5.17	vbat_ldo	488
25.5.18	rc32k_ctrl0	490
25.5.19	xtal32k	492
26	SEC ENG	493
26.1	Overview	493
26.1.1	AES	493
26.1.2	SHA	493
26.1.3	CRC	493
26.1.4	GMAC	493
26.2	Features	494
26.3	Principle	494
26.3.1	AES encryption and decryption process	494
26.3.2	Implementation of SHA256	495
26.3.3	Principle of GMAC	496
26.4	Functional Description	498
26.4.1	AES Accelerator	498
26.4.2	SHA Accelerator	499
26.4.3	SHA Accelerator	499
26.4.4	GMAC(link Mode)	500
26.4.5	Random Number Generator (RNG)	501
27	Revision history	502

List of Figures

1.1 System architecture	27
2.1 System Clock Architecture	36
2.2 Module Clock Architecture	37
2.3 Peripheral Clock Architecture	38
5.1 Block Diagram of ADC	130
5.2 ADC Clocks	132
6.1 Block Diagram of DAC	140
7.1 Block Diagram of DMA	148
7.2 Peripheral Type Selection	150
7.3 LLI Framework	152
8.1 NEC Logic Waveform	178
8.2 NEC Protocol Waveform	178
8.3 RC5 Logic Waveform	178
8.4 RC5 Protocol Waveform	179
9.1 SPI Timing	186
9.2 SPI Ignore Waveform	187
9.3 SPI Filter Waveform	188
10.1 UART Data Format	201
10.2 UART Architecture	202
10.3 UART Sampling Waveform	203
10.4 UART Filter Waveform	204
10.5 Waveform of UART in fixed character mode	205
10.6 UART hardware flow control	205

10.7 A typical LIN frame	207
10.8 Break Field of LIN	207
10.9 Sync Field of LIN	207
10.10 ID Field of LIN	208
11.1 Start and stop conditions of I2C	230
11.2 I2C data transfer format	231
11.3 Timing of master transmitter and slave receiver	231
11.4 Timing of master receiver and slave transmitter	231
11.5 Waveform of simultaneous data transfer	233
12.1 Waveform of PWM in different configurations	248
13.1 Block diagram of timer	261
13.2 Block diagram of watchdog timer	262
13.3 Working sequence of timer in preLoad mode	264
13.4 Working sequence of watchdog	265
13.5 Watchdog alarm mechanism	266
14.1 Normal I2S	286
14.2 I2S LeftJustified/RightJustified	288
14.3 I2S TDM64 Mode—6-Channel Recording	289
14.4 I2S clock	289
15.1 Block Diagram of Clock	298
15.2 Block Diagram of Module	298
15.3 Mixer	301
16.1 Block Diagram of Clock	316
16.2 Block Diagram of Module	316
17.1 Block diagram of EMAC	338
19.1 Single filter configuration, receiving standard frame messages	351
19.2 Single filter configuration, receiving extended frame messages	352
19.3 Dual filter configuration, receiving standard frame messages	353
19.4 Dual filter configuration, receiving extended frame messages	354
19.5 Arbitration lost bit number interpretation	355
19.6 Example of arbitration lost bit number interpretation; result: ALC = 08	355
19.7 General structure of a bit period	359
20.1 CAM block diagram	361
20.2 FIFO framework	364
20.3 Memory	365

22.1 DBI basic block diagram	383
22.2 Write timing	384
22.3 Read timing	385
22.4 RGB565 output	386
22.5 RGB666 output	387
22.6 RGB888 output	388
22.7 Write timing	389
22.8 Read timing	389
22.9 RGB565 output	390
22.10 RGB666 output	390
22.11 RGB888 output	391
22.12 Write timing	391
22.13 Read timing	392
22.14 RGB565 output	393
22.15 RGB666 output	393
22.16 RGB888 output	394
22.17 Write timing	395
22.18 Read timing	396
22.19 RGB565 output	397
22.20 RGB666 output	397
22.21 RGB888 output	398
22.22 RGB565 output	399
22.23 RGB666 output	400
22.24 RGB888 output	401
22.25 RGB565 output	401
22.26 RGB666 output	402
22.27 RGB888 output	402
22.28 Input Pixel RGB Format	403
23.1 SDH Hardware and Driver Structure	406
23.2 Standard Register Mapping Classification	407
23.3 Register Mapping of Multi-Card Slot Controller	408
23.4 Register for Generating SD Commands	409
23.5 Suspend and Resume Mechanism	410
23.6 Block Diagram of Host Controller	412
23.7 Controlling SDCLK by the SD Bus Power and SD Clock Enable	413
23.8 Block Diagram of ADMA2	415
23.9 32-Bit Address Descriptor Table	417
23.10 ADMA2 States	418
23.11 SD Host Control Register Mapping	421
23.12 Types of Registers and Register Bit Fields	422

24.1 Signal Connection of Two 4-Bit SDIO Cards	425
24.2 SDIO's Response to Non-I/O Aware Initialization	427
24.3 IO_SEND_OP_COND Command	428
24.4 Response R4 in SD Mode	428
24.5 Response R4 in SPI Mode	429
24.6 Modified R1 Response	429
24.7 Command List for SD Mode	431
24.8 Command List for SPI Mode	432
24.9 Unsupported SD Memory Commands	433
24.10 R6 Response of CMD3	434
24.11 SDIO R6 Status Bit	434
24.12 IO_RW_DIRECT Command	436
24.13 IO_RW_DIRECT Response in SD Mode	437
24.14 IO_RW_DIRECT Response in SPI Mode	438
24.15 IO_RW_EXTENDED Command	438
24.16 SDIO Internal Mapping	441
24.17 State Diagram for Bus State Machine	446
25.1 Low power modes	450
26.1 AES Encryption Process	494
26.2 MAC Flow Chart	497
26.3 AES Operation Modes	498

List of Tables

1.1 Boot mode	29
1.2 Address mapping	29
1.3 Interrupt assignment	30
1.4 Peripheral List	32
2.1 Software reset function table	34
4.1 GPIO function list	41
5.1 ADC internal signal	131
5.2 ADC external pin	131
5.3 Meaning of ADC Conversion Result	134
6.1 Data transfer format	141
6.2 Internal reference voltage output voltage	141
11.1 I2C pin list	230
14.1 I2S pin list	286
17.1 Transmission signal	340
19.1 The meaning of each register in different modes	347
19.2 Arbitration loss capture location	355
19.3 Type of error catch	357
19.4 Error catch location	357
25.1 Power mode	453
25.2 Wakeup source	453
27.1 Document revision history	502

System and Memory

1.1 System Architecture

BL616/BL618 series chips adopt RISC-V 32-bit CPU, equipped with 16KB D-cache and 32KB I-Cache, CPU frequency up to 320MHz, suitable for high-performance applications such as Internet of Things, embedded and artificial intelligence.

The main system consists of the following parts:

- Bus interface: AXI bus and AHB bus
 - CPU accesses memory through AXI bus
 - The CPU accesses peripherals through the AHB bus
 - With low power consumption, high performance and other characteristics
- memory
 - 480KB SRAM for data storage
 - 128KB ROM
 - Support embedded high-speed PSRAM, expand RAM capacity
- Peripherals
 - A total of 30 peripheral modules, including advanced peripherals such as USB/SDH/SDU/EMAC/DVP/Display

The Wi-Fi 6/BLE/Zigbee wireless subsystem is integrated on-chip, which can realize a variety of wireless connections and data transmission, and provide a variety of connection and transmission experiences.

The system architecture of BL616/BL618 is shown in the following figure:

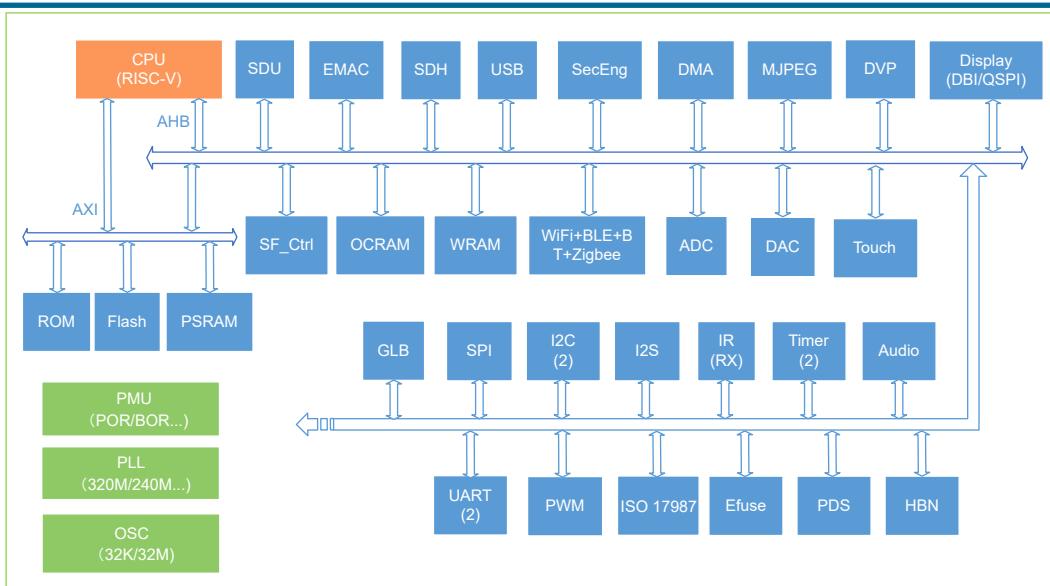


Fig. 1.1: System architecture

1.2 CPU

1.2.1 Features

BL616/BL618 has a built-in 32-bit RISC-V CPU, uses a 5-stage pipeline: fetch, decode, execute, memory access, and write back. This high-performance embedded microprocessor has the following features:

- 32-bit RISC processor
- Supports RISC-V RV32IMAFCP instruction set
- Supports RISC-V 32-bit/16-bit mixed instruction set
- Supports RISC-V machine mode and user mode
- Thirty-two 32-bit integer general purpose registers (GPR) and thirty-two 32-bit/64-bit floating-point GPRs
- Integer (5-stage)/floating-point (7-stage), single-issue, sequentially executed pipeline
- Supports AXI 4.0 main device interface and AHB 5.0 peripheral interface
- 32KB instruction cache, two-way set associative structure
- 16KB data cache, two-way set associative structure
- Unaligned memory access
- Double-cycle hardware multiplier and Radix-4 hardware divider
- Supports BHT (8K) and BTB
- Supports extended enhanced instruction set

- Supports MCU feature extension technique, including interrupt processing acceleration technique and MCU extension feature
- Compatible with RISC-V CLIC interrupt standard and interrupt nesting; 96 external interrupt sources, 4 bits for configuring interrupt priority
- Compatible with RISC-V PMP, 8 configurable areas
- Supports hardware performance monitor (HPM) units
- Supports RISC-V Debug Specification

1.2.2 Extended Functions

- Bit operation instruction, arithmetic operation instruction, and enhanced memory access instruction
- CACHE operation instruction and synchronization instruction, making programming easy for software programmers
- Interrupt speculative push technique and vector interrupt tail-biting technique, accelerated interrupt response, with interrupt response delay of 20 processor clock cycles
- Common application requirements in MCU fields like NMI, low-power mode for deep/light sleep, low-power wake-up events, and locking
- Supports unaligned memory access, which can be enabled/disabled by software, making programming and debugging easy for software programmers

1.2.3 Implemented Standards

The processor is compatible with the RISC-V standard and the specific versions are:

- The RISC-V Instruction Set Manual, Volume I: RISC-V User-Level ISA, Version 2.2
- The RISC-V Instruction Set Manual, Volume II: RISC-V Privileged Architecture, Version 1.10
- RISC-V Core-Local Interrupt Controller (CLIC) Version 0.8
- RISC-V External Debug Support Version 0.13.2
- RISC-V “P” Extension Proposal Version 0.9

1.3 Boot

The system supports boot from Flash/UART/USB/SDU, as described below:

Table 1.1: Boot mode

Boot Pin	Level	Description
GPIO2	1	Boot from UART(GPIO21/22)/USB/SDU, this mode is mainly used for flash programming or downloading image to RAM for execution (wireless transparent transmission scenario)
	0	Boot the application image from Flash

1.4 Address Mapping

Table 1.2: Address mapping

Module	Target	Base Address	Size	Description
FLASH	Flash	0xA0000000	128MB	Application address space
PSRAM	pSRAM	0xA8000000	128MB	pSRAM memory address space (optional, depends on the specific chip model)
RAM	OCRAM	0x20FC0000	320KB	On Chip RAM address space, mainly used for CPU application data
	WRAM	0x21010000	160KB	Wireless RAM address space, mainly used for wireless network data
	HBN RAM	0x20010000	4KB	HBN RAM,mainly used for data saving in ultra-low power mode
Peripheral	USB	0x20072000	4KB	USB High Speed OTG Control Register
	EMAC	0x20070000	4KB	EMAC Control Register
	SDH	0x20060000	4KB	SDH Control Register
	MJPEG	0x20059000	4KB	MJPEG Control Register
	DVP	0x20057000	4KB	DVP camera interface Control Register
	Efuse	0x20056000	4KB	Efuse storage Control Register
	AUDIO PWM	0x20055000	4KB	Audio PWM Control Register
	PSRAM_Ctrl	0x20052000	4KB	PSRAM Control Register
	HBN	0x2000F000	4KB	Hibernate register
	PDS	0x2000E000	4KB	Power-down sleep register
	SDU	0x2000D000	4KB	SDU Control Register
	DMA	0x2000C000	4KB	DMA Control Register
	SF_Ctrl	0x2000B000	4KB	Serial Flash Control Register
	Audio ADC	0x2000AC00	256B	Audio ADC Control Register

Table 1.2: Address mapping (continued)

Module	Target	Base Address	Size	Description
Peripheral	I2S	0x2000AB00	256B	I2S Control Register
	ISO 17987	0x2000AA00	256B	ISO 17987 Control Register
	I2C1	0x2000A900	256B	I2C1 Control Register
	Display	0x2000A800	256B	Display Control Register
	IRR	0x2000A600	256B	IR Receiver Control Register
	TIMER	0x2000A500	256B	TIMER Control Register
	PWM	0x2000A400	256B	PWM Control Register
	I2C0	0x2000A300	256B	I2C0 Control Register
	SPI	0x2000A200	256B	SPI Control Register
	UART1	0x2000A100	256B	UART1 Control Register
	UART0	0x2000A000	256B	UART0 Control Register
	TZ	0x20005000	4KB	TrustZone Control Register
	SEC_ENG	0x20004000	4KB	Security Engine Control Register
	GPIP	0x20002000	1KB	General Purpose DAC/ADC/ACOMP Interface Control Register
	GLB	0x20000000	4KB	Global control register
ROM	ROM	0x90000000	128KB	Bootrom address space

1.5 Interrupt Source

A total of 64 interrupt sources are included, and the interrupt sources and corresponding interrupt numbers are shown in the following table:

Table 1.3: Interrupt assignment

Interrupt source		Interrupt Number	Description
BMX	BUS Error	IRQ_NUM_BASE+0	BUS Error Responses Interrupt
	BUS Timeout	IRQ_NUM_BASE+1	BUS Responses Timeout Interrupt
Dispaly	Dispaly	IRQ_NUM_BASE+2	Dispaly All Interrupt
SDU	SDU Software Reset	IRQ_NUM_BASE+3	SDU Reset Triggered by Host
Audio	Audio	IRQ_NUM_BASE+4	Audio All Interrupt
RF	RF Interrupt0	IRQ_NUM_BASE+5	RF Interrupt0
	RF Interrupt1	IRQ_NUM_BASE+6	RF Interrupt1
SDU	SDU Side Interrupt	IRQ_NUM_BASE+7	SDU Side All Interrupt
WiFi	WiFi TBTT	IRQ_NUM_BASE+8	WiFi TBTT Interrupt
	Group0	IRQ_NUM_BASE+9	Group0 SHA/AES/TRNG/PKA/GMAC Interrupt

SecEng

Table 1.3: Interrupt assignment (continued)

Interrupt source		Interrupt Number	Description
	Group1	IRQ_NUM_BASE+10	Group1 SHA/AES/TRNG/PKA/GMAC Interrupt
	Group0 CDET	IRQ_NUM_BASE+11	Group0 CDET Interrupt
	Group1 CDET	IRQ_NUM_BASE+12	Group1 CDET Interrupt
SF Ctrl	Group0	IRQ_NUM_BASE+13	SF_Ctrl Group0 Interrupt
	Group1	IRQ_NUM_BASE+14	SF_Ctrl Group1 Interrupt
DMA	DMA0_ALL	IRQ_NUM_BASE+15	DMA0 ALL Interrupt
DVP0	DVP2BUS0	IRQ_NUM_BASE+16	DVP2BUS0 Interrupt
SDH	SDH All Interrupt	IRQ_NUM_BASE+17	SDH All Interrupt
DVP1	DVP2BUS1	IRQ_NUM_BASE+18	DVP2BUS1 Interrupt
WiFi	TBTT	IRQ_NUM_BASE+19	WiFi TBTT Interrupt
IR	IRRX	IRQ_NUM_BASE+20	IR RX Interrupt
USB	USB	IRQ_NUM_BASE+21	USB Interrupt
Audio	Record	IRQ_NUM_BASE+22	Audio Record All Interrupt
MJPEG	Encoder	IRQ_NUM_BASE+23	MJPEG Encoder All Interrupt
EMAC	EMAC	IRQ_NUM_BASE+24	EMAC Interrupt
ADC	GPADC_DMA	IRQ_NUM_BASE+25	GPADC_DMA Interrupt
Efuse	Efuse	IRQ_NUM_BASE+26	Efuse Interrupt
SPI	SPI	IRQ_NUM_BASE+27	SPI Interrupt
UART	UART0	IRQ_NUM_BASE+28	UART0 Interrupt
	UART1	IRQ_NUM_BASE+29	UART1 Interrupt
ISO 17987	ISO 17987	IRQ_NUM_BASE+30	ISO 17987 Interrupt
GPIO	GPIO_DMA	IRQ_NUM_BASE+31	GPIO DMA Interrupt
I2C0	I2C0	IRQ_NUM_BASE+32	I2C0 Interrupt
PWM	PWM	IRQ_NUM_BASE+33	PWM Interrupt
TIMER0	TIMER0_CH0	IRQ_NUM_BASE+36	Timer0 Channel 0 Interrupt
	TIMER0_CH1	IRQ_NUM_BASE+37	Timer0 Channel 1 Interrupt
	TIMER0_WDT	IRQ_NUM_BASE+38	Timer0 Watch Dog Interrupt
I2C1	I2C1	IRQ_NUM_BASE+39	I2C1 Interrupt
I2S	I2S	IRQ_NUM_BASE+40	I2S Interrupt
	Reserved	IRQ_NUM_BASE+41	Reserved
	Reserved	IRQ_NUM_BASE+42	Reserved
XTAL	Xtal Ready	IRQ_NUM_BASE+43	Xtal Ready Interrupt
GPIO	GPIO_INT0	IRQ_NUM_BASE+44	GPIO Interrupt
DM	DM	IRQ_NUM_BASE+45	DM Interrupt
BT	BT	IRQ_NUM_BASE+46	BT Interrupt
	ENH Ack	IRQ_NUM_BASE+47	MAC154 ENH Ack Interrupt

MAC154

Table 1.3: Interrupt assignment (continued)

Interrupt source		Interrupt Number	Description
	Others	IRQ_NUM_BASE+48	MAC154 Other Interrupt
	AES	IRQ_NUM_BASE+49	MAC154 AES Interrupt
PDS	PDS	IRQ_NUM_BASE+50	PDS Interrupt
HBN	HBN OUT0	IRQ_NUM_BASE+51	HBN Out 0 Interrupt
	HBN OUT1	IRQ_NUM_BASE+52	HBN Out 1 Interrupt
BOD	BOD	IRQ_NUM_BASE+53	Break Out Detect Interrupt
WiFi	WiFi	IRQ_NUM_BASE+54	WiFi Interrupt
BZ Phy	BZ Phy	IRQ_NUM_BASE+55	BZ Phy Interrupt
BLE	BLE	IRQ_NUM_BASE+56	BLE Interrupt
WiFi	MAC TR Timer	IRQ_NUM_BASE+57	MAC TX&RX Timer Interrupt
	MAC TR MISC	IRQ_NUM_BASE+58	MAC TX&RX Misc Interrupt
	MAC RX Trigger	IRQ_NUM_BASE+59	MAC RX Trigger Interrupt
	MAC TX Trigger	IRQ_NUM_BASE+60	MAC TX Trigger Interrupt
	MAC General	IRQ_NUM_BASE+61	MAC General Interrupt
	MAC Prot	IRQ_NUM_BASE+62	MAC Prot Interrupt
	IPC	IRQ_NUM_BASE+63	MAC IPC Interrupt

Note: IRQ_NUM_BASE is 16 and the interrupt number 015 is RISC-V reserved interrupt.

1.6 Peripheral Overview

Table 1.4: Peripheral List

Peripheral	Number	Note
GPIO	19/35	QFN40 corresponds to 19 GPIOs, QFN56 corresponds to 35 GPIOs
UART	2	Support RTS/CTS
SPI	1	Support Master/Slave mode
I2C	2	Support Master mode
ISO11898	1	AHB bus
I2S	1	Support Left-Justified/Right-Justified/Normal I2S/DSP and other data formats
PWM	4	Support adjustable output polarity, dual threshold value setting
Timer	2	Support FreeRun mode and PreLoad mode

Table 1.4: Peripheral List (continued)

Peripheral	Number	Note
DMA	4	Support LLI linked list function
IR	1	Support receiving, the protocol includes NEC and RC-5, and also supports receiving data by pulse width counting
Audio PWM	1	Support audio playback
Audio ADC	1	Support recording
EMAC	1	Supports 10Mbps and 100Mbps
CAM	2	Support image rectangle crop
MJPEG	1	Supports arbitrary quantization tables
DBI	1	Support Type B/Type C 3-wire/Type C 4-wire, and also integrates QSPI mode
SDH	1	Support high-speed SD card
SDU	1	Support CCCR (function0) and function1, support SDU soft reset, function1 has 16 port receive buffers
SEC_ENG	1	Support AES/SHA/GMAC/TRNG
USB	1	USB2.0

Reset and Clock

2.1 Overview

The clock sources in the chip include XTAL, PLL, and RC. They are sent to each module along with frequency division configuration.

2.2 Reset Management

Table 2.1: Software reset function table

BL616	RST_PIN/ Watch Dog/ PDS/ Software Power On (swrst_cfg2[0])	Software Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])/ PDS	CPU Reset (swrst_cfg2[1])/ PDS
cpu(M0)	✓			✓
bus	✓		✓	
glb	✓	swrst_s1[0]		
mix	✓	swrst_s1[1]		
gpip	✓	swrst_s1[2]	✓	
sec_eng	✓	swrst_s1[4]	✓	
tz	✓		✓	
efuse	✓			
dma	✓	swrst_s1[12]	✓	
sdu/usb	✓	swrst_s1[13]		
mm_misc	✓	swrst_s1_ext[1]	✓	
psram_ctrl	✓	swrst_s1_ext[2]	✓	
usb	✓	swrst_s1_ext[3]	✓	
audio play	✓	swrst_s1_ext[5]	✓	
sdh	✓	swrst_s1_ext[6]	✓	
emac	✓	swrst_s1_ext[7]	✓	

Table 2.1: Software reset function table(continued)

BL616	RST_PIN/ Watch Dog/ PDS/ Software Power On (swrst_cfg2[0])	Software Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])/ PDS	CPU Reset (swrst_cfg2[1])/ PDS
dma2	✓	swrst_s1_ext[8]	✓	
pds		swrst_s1[14]		
uart0	✓	swrst_s1a[0]	✓	
uart1	✓	swrst_s1a[1]	✓	
spi	✓	swrst_s1a[2]	✓	
i2c0	✓	swrst_s1a[3]	✓	
pwm	✓	swrst_s1a[4]	✓	
timer	✓	swrst_s1a[5]	✓	
irr	✓	swrst_s1a[6]	✓	
cks	✓	swrst_s1a[7]	✓	
dbi	✓	swrst_s1a[8]	✓	
i2c1	✓	swrst_s1a[9]	✓	
ISO11898_- fd	✓	swrst_s1a[10]	✓	
i2s	✓	swrst_s1a[11]	✓	
audio record	✓	swrst_s1a[12]	✓	
wifi	✓	swrst_s2[0]		
ble	✓	swrst_s3[0][2]		

2.3 Clock Source

Types:

- XTAL: external crystal oscillator clock, with optional frequencies of 24, 32, 38.4, and 40 MHz depending on system requirements
- XTAL32K: external crystal oscillator clock, with frequency of 32 kHz
- RC32K: RC oscillator clock with frequency of 32 kHz and calibration
- RC32M: RC oscillator clock with frequency of 32 MHz and calibration
- PLL: multiple PLL modules, which can generate several clocks with different frequencies to meet various application scenarios

The clock control unit distributes the clocks from the oscillator to the core and peripheral devices. You can choose

the system clock source, dynamic frequency divider, and clock configuration, and use the 32 kHz clock in sleep to achieve low-power clock management.

Peripheral clocks include Flash, UART, I2C, SPI, PWM, IR-remote, ADC, and DAC.

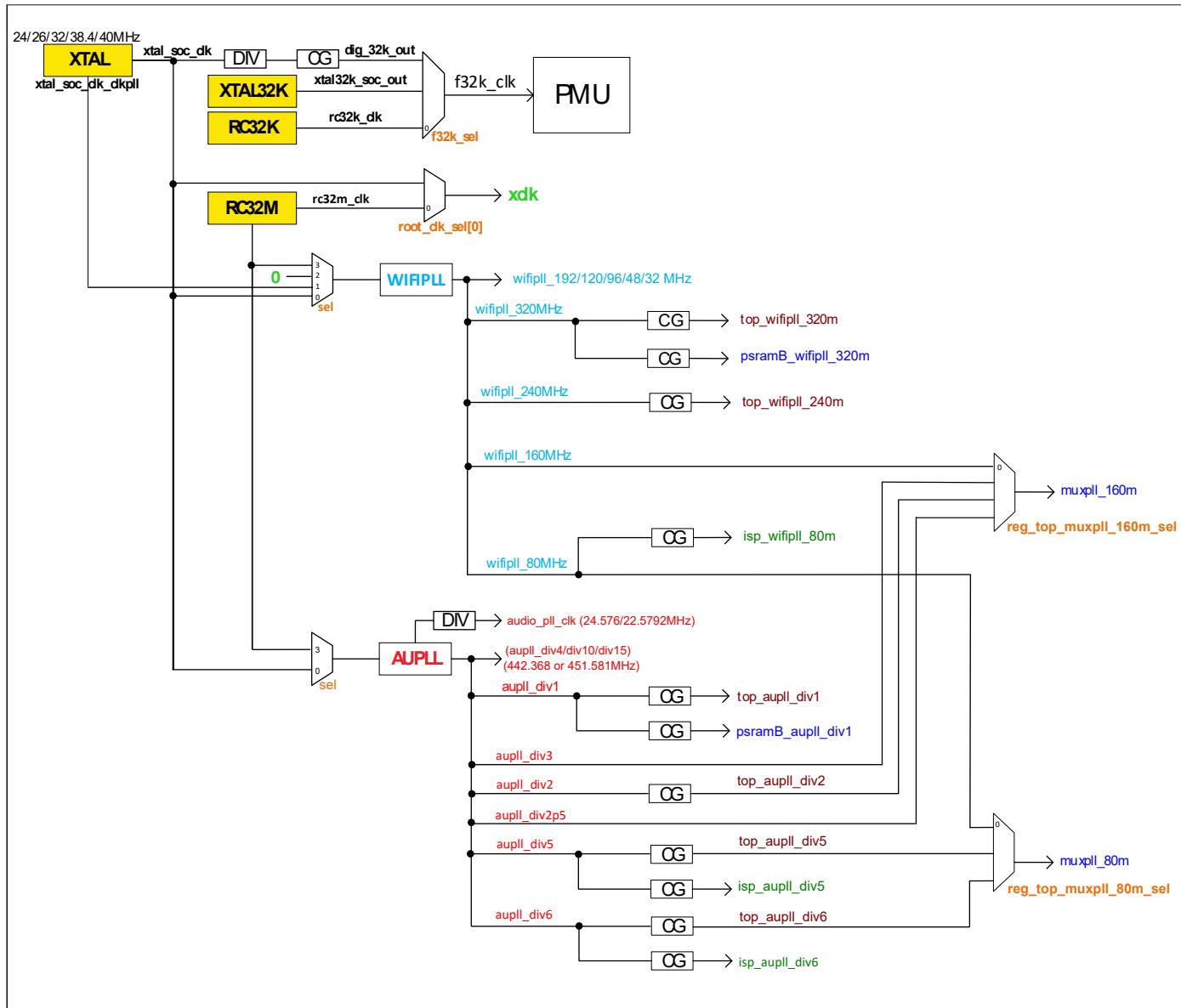


Fig. 2.1: System Clock Architecture

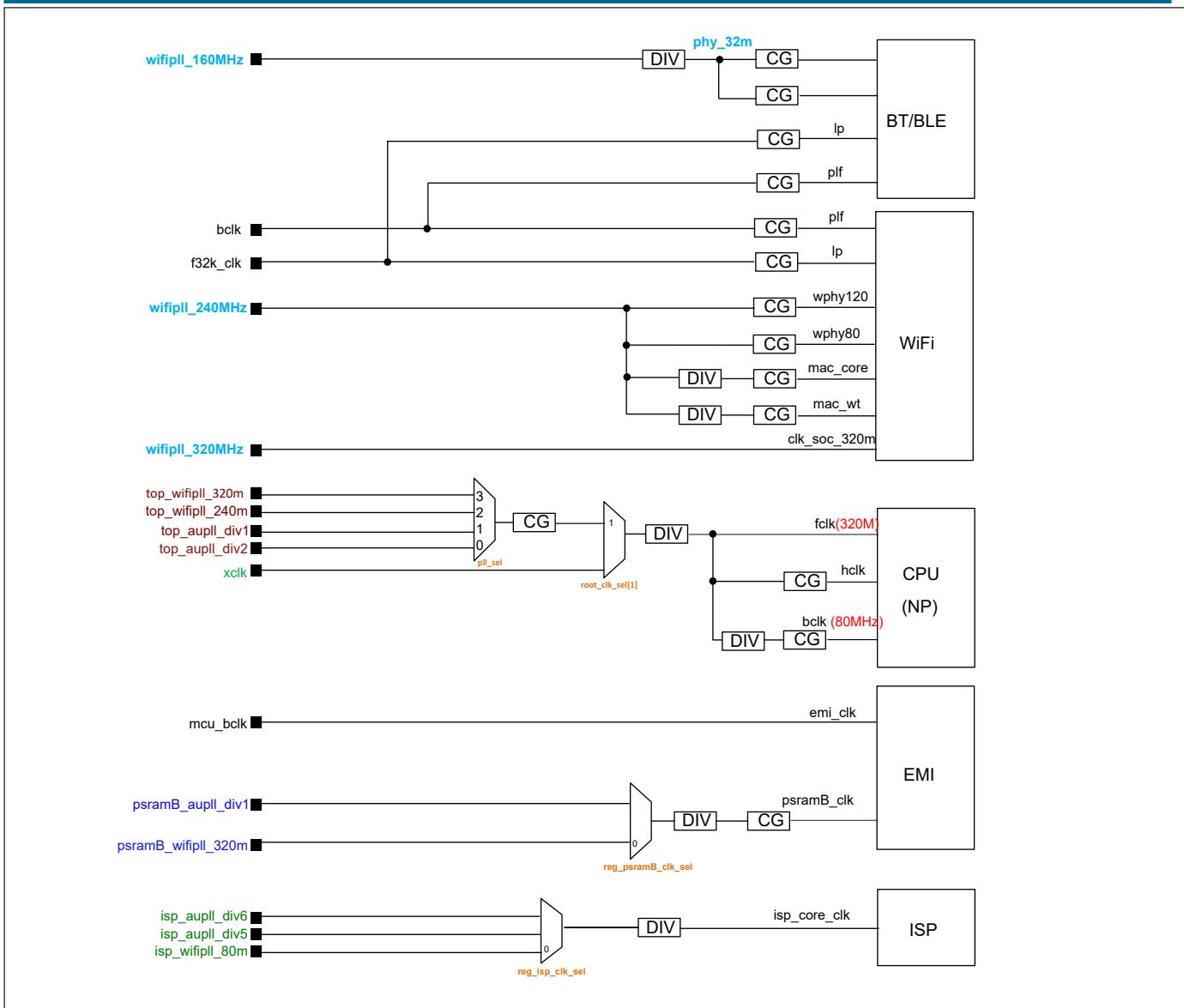


Fig. 2.2: Module Clock Architecture

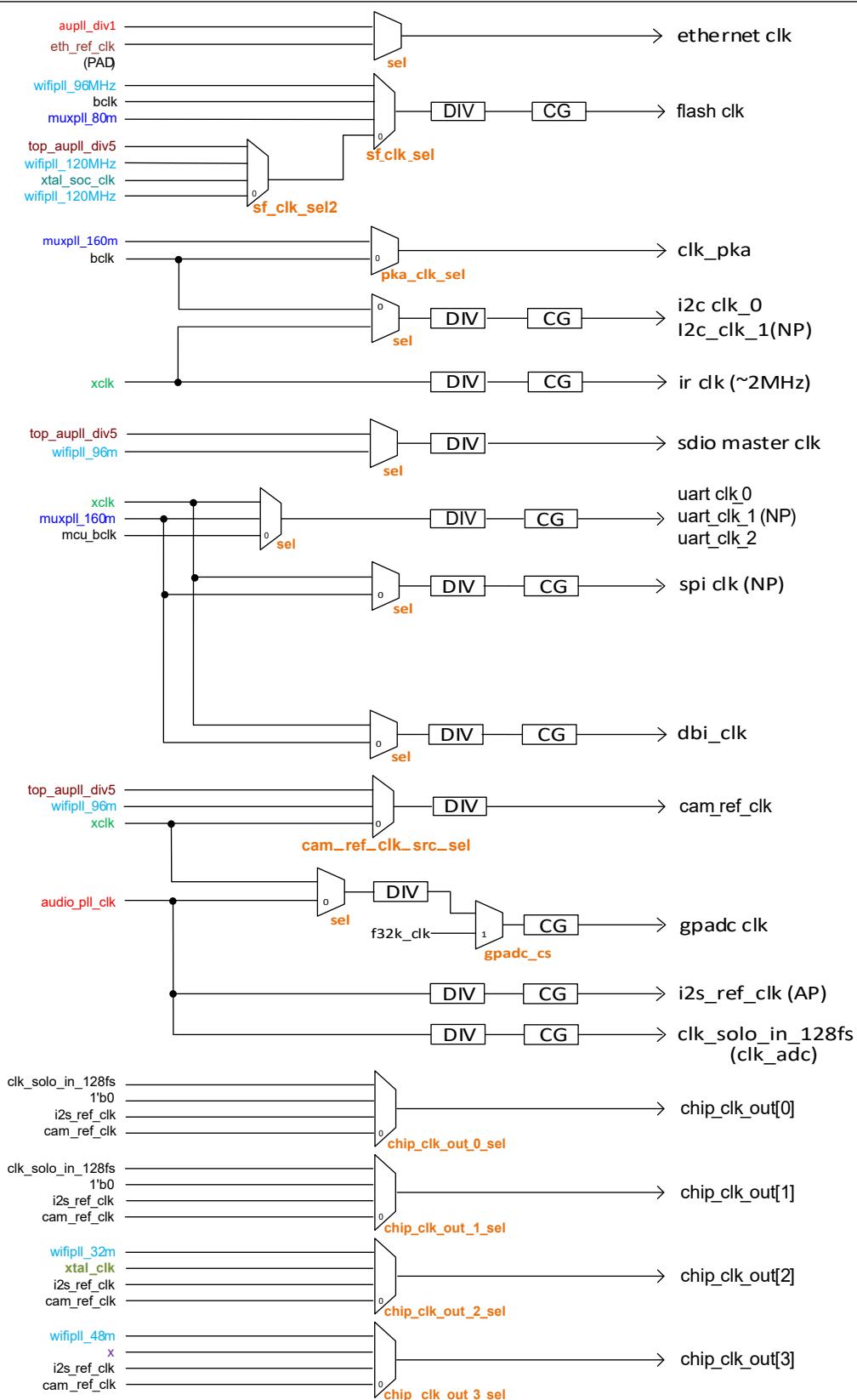


Fig. 2.3: Peripheral Clock Architecture

3.1 Overview

Global Register (GLB), a general global setting module for chips, mainly includes clock, reset, bus, memory, LDO, and GPIO management.

3.2 Functional Description

3.2.1 Clock Management

It is mainly used to set the clocks of processors, buses, and peripherals, set the clock source and frequency division of the above modules, and achieve the clock gating of these modules, to save power for the system. For more details, see system clock sections.

3.2.2 Reset Management

It provides the separate reset function of each peripheral module and the chip reset function.

Chip reset:

- CPU reset: Only CPU is reset. The program will roll back, and peripherals will not be reset
- System reset: All peripherals and CPUs will be reset, but the registers in the AON field will not be reset
- Power-on reset: The whole system including the registers in the AON field will be reset

The application can select a proper reset mode as required.

3.2.3 Bus Management

It provides bus arbitration and bus error settings, so that users can set whether to interrupt and provide the error bus address information when a bus error occurs, facilitating program debugging by users.

3.2.4 Memory Management

It manages the size of each memory area:

- EM management: A part of space of WRAM serves as EM, 32 KB by default. This function can assign the space of a specified size to EM, while the remaining one serves as WRAM.

It provides power management of each memory module in the low-power mode of the chip system, including two setting modes:

- Retention mode: The data in the memory can be saved, but it cannot be read or written before exiting the low-power mode
- Sleep mode: This mode is only used to reduce system power consumption, because it will cause memory data to lose

3.2.5 LDO Management

It provides on-chip LDO management:

- LDO management: The output voltage of the on-chip LDO can be adjusted to reduce power consumption.

4.1 Overview

Users can connect General Purpose I/O Ports (GPIO) with external hardware devices to control these devices.

4.2 Features

- Up to 35 I/O pins
- Each I/O pin supports up to 25 functions
- Each I/O pin can be configured in pull-up, pull-down, or floating mode
- Each I/O pin can be configured as input, output or Hi-Z state mode
- The output mode of each I/O pin has 4 optional drive capabilities
- The input mode of each I/O pin can be set to enable/disable the Schmitt trigger
- Each I/O pin supports 9 external interrupt modes
- FIFO depth: 128 halfwords
- Data can be transferred through DMA from RAM to I/O pins for output

4.3 GPIO function list

The <reg_gpio_xx_func_sel> bit of the GPIO_CFGxx (where xx represents the GPIO pin number) register is used to set the multiplexing function of GPIO. The multiplexing function number is shown in the following table:

Table 4.1: GPIO function list

Number	Function
0	SDH

Table 4.1: GPIO function list (continued)

Number	Function
1	SPI0
2	FLASH
3	I2S0
4	PDM
5	I2C0
6	I2C1
7	UART0
8	EMAC
9	CAM
10	ANALOG
11	GPIO
12	SDIO
16	PWM0
17	JTAG
18	UART1
19	PWM1
20	SPI1
21	I2S1
22	DBI_B
23	DBI_C
24	QSPI
25	AUPWM
31	CLOCK_OUT

Note: If the GPIO is set as a peripheral function, just set the <reg_gpio_xx_func_sel> bit of the GPIO_CFGxx register to the corresponding number of the peripheral.

4.4 GPIO Input Settings

Set the register GPIO_CFGxx to configure the general interface to the input mode as described below (xx denotes the GPIO pin number):

- Set `<reg_gpio_xx_ie>` to 1 to enable the GPIO input mode
- Set `<reg_gpio_xx_func_sel>` to 11 to enter the SWGPIO mode
- In the SWGPIO mode, set to enable/disable the Schmitt trigger through `<reg_gpio_xx_smt>` for waveform shaping
- Set to enable/disable the internal pull-up and pull-down functions through `<reg_gpio_xx_pu>` and `<reg_gpio_xx_pd>`
- Set the type of external interrupt through `<reg_gpio_xx_int_mode_set>`, and then read the level value of I/O pin through `<reg_gpio_xx_i>`

4.5 GPIO Output Settings

The following four output modes of GPIO can be set through the register GPIO_CFGxx.

4.5.1 Normal Output Mode

- Set `<reg_gpio_xx_oe>` to 1 to enable the GPIO output mode
- Set `<reg_gpio_xx_func_sel>` to 11 to enter the SWGPIO mode
- Set `<reg_gpio_xx_mode>` to 0 to enable the normal output function of I/O
- Set to enable/disable the internal pull-up and pull-down functions through `<reg_gpio_xx_pu>` and `<reg_gpio_xx_pd>`, and then set the level of I/O pin through `<reg_gpio_xx_o>`

4.5.2 Set/Clear Output Mode

- Set `<reg_gpio_xx_oe>` to 1 to enable the GPIO output mode
- Set `<reg_gpio_xx_func_sel>` to 11 to enter the SWGPIO mode
- Set `<reg_gpio_xx_mode>` to 1 to enable the Set/Clear output function of I/O
- Set to enable/disable the internal pull-up and pull-down functions through `<reg_gpio_xx_pu>` and `<reg_gpio_xx_pd>`

In the Set/Clear output mode, you can set `<reg_gpio_xx_set>` to 1 to keep the I/O pin at the high level, or set `<reg_gpio_xx_clr>` to 1 to keep the I/O pin at the low level. If both `<reg_gpio_xx_set>` and `<reg_gpio_xx_clr>` are set to 1, the I/O pin is kept at the high level. If both of them are set to 0, the setting does not work.

4.6 I/O FIFO

The depth of I/O FIFO is 128 halfwords. The `<gpio_tx_fifo_cnt>` bit in the register GPIO_CFG143 indicates the current available space of FIFO (128 by default). Every time a value is written into the GPIO_CFG144 register, the value of `<gpio_tx_fifo_cnt>` will decrease by 1. After it decreases to 0, if a value is continuously written to the register GPIO_CFG144 and `<cr_gpio_tx_fer_en>` is 1, the error interrupt will be enabled and this interrupt will occur.

When the `<cr_gpio_tx_en>` bit in the GPIO_CFG142 register is 1, the data of I/O FIFO will be sent to I/O pins one by one, and the value of `<gpio_tx_fifo_cnt>` will increment. When it is incremented to greater than `<cr_gpio_tx_fifo_th>` and `<cr_gpio_tx_fifo_en>` is 1, the FIFO interrupt is enabled and this interrupt will occur.

If the `<cr_gpio_dma_tx_en>` bit in the register CR_GPIO_CFG143 is 1, DMA is enabled to send data. If `<cr_gpio_tx_fifo_th>` is less than `<gpio_tx_fifo_cnt>`, DMA will transfer the data from the preset RAM to the buffer, whereas the interrupt flag `<r_gpio_tx_fifo_int>` will be cleared automatically.

4.7 I/O Interrupt

I/O supports various external interrupts. Setting the `<reg_gpio_xx_int_mask>` in the register GPIO_CFGxx to 0 can enable the external interrupt of the corresponding pin. `<reg_gpio_xx_int_mode_set>` is used to set the external interrupt type of that pin.

The supported interrupt types are as follows:

- Synchronous Falling Edge Interrupt
 - Based on the f32k_clk clock, the input pin level is sampled once on each rising edge of the clock. If a high level is followed by two low levels, a synchronous falling edge interrupt will be generated at this time
- Synchronous Rising Edge Interrupt
 - Based on the f32k_clk clock, the input pin level is sampled once on each rising edge of the clock. If a low level is followed by two high levels, a synchronous rising edge interrupt will be generated at this time
- Synchronous Low Level Interrupt
 - Based on the f32k_clk clock, after detecting a low level, a synchronous low-level interrupt is generated at the rising edge of the third clock
- Synchronous High Level Interrupt
 - Based on the f32k_clk clock, after detecting a high level, a synchronous high-level interrupt is generated at the rising edge of the third clock
- Synchronous Double Edge Interrupt
 - Based on the f32k_clk clock, if a high level transition to low level (low level transition to high level) is detected, a falling edge (rising edge) event will be generated. After the event is generated, at the third rising edge of the clock, synchronous double edge interrupt will be generated

- Asynchronous Falling Edge Interrupt
 - When a high-to-low transition is detected, an asynchronous falling edge interrupt is triggered immediately
- Asynchronous Rising Edge Interrupt
 - When a low-to-high transition is detected, an asynchronous rising edge interrupt is triggered immediately
- Asynchronous Low Level Interrupt
 - Based on the f32k_clk clock, the input pin level is sampled once on each rising edge of the clock. If it is low for 3 consecutive times, an asynchronous low-level interrupt is triggered
- Asynchronous High Level Interrupt
 - Based on the f32k_clk clock, the input pin level is sampled once on each rising edge of the clock. If it is high for 3 consecutive times, an asynchronous high-level interrupt will be triggered

In the interrupt function, you can obtain the interrupt-generating GPIO state through the <gpio_xx_int_stat> of the register GPIO_CFGxx, and clear the interrupt flag through <reg_gpio_xx_int_clr>.

4.8 Register description

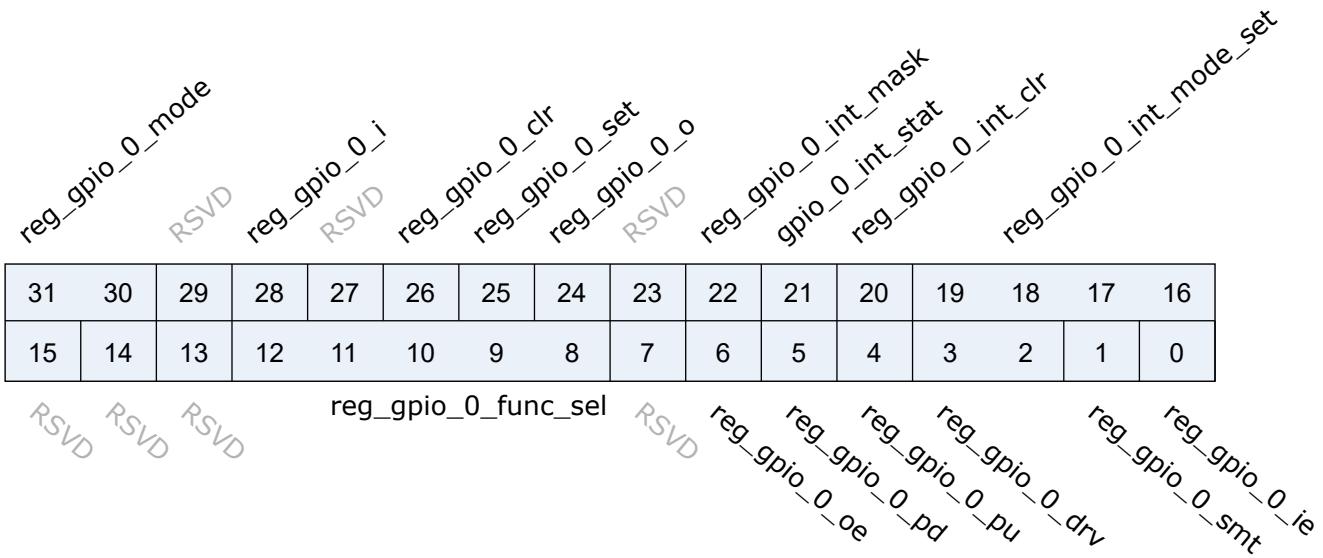
Name	Description
gpio_cfg0	
gpio_cfg1	
gpio_cfg2	
gpio_cfg3	
gpio_cfg4	
gpio_cfg5	
gpio_cfg6	
gpio_cfg7	
gpio_cfg8	
gpio_cfg9	
gpio_cfg10	
gpio_cfg11	
gpio_cfg12	
gpio_cfg13	
gpio_cfg14	

Name	Description
gpio_cfg15	
gpio_cfg16	
gpio_cfg17	
gpio_cfg18	
gpio_cfg19	
gpio_cfg20	
gpio_cfg21	
gpio_cfg22	
gpio_cfg23	
gpio_cfg24	
gpio_cfg25	
gpio_cfg26	
gpio_cfg27	
gpio_cfg28	
gpio_cfg29	
gpio_cfg30	
gpio_cfg31	
gpio_cfg32	
gpio_cfg33	
gpio_cfg34	
gpio_cfg128	
gpio_cfg129	
gpio_cfg136	
gpio_cfg137	
gpio_cfg138	
gpio_cfg139	
gpio_cfg141	
gpio_cfg142	
gpio_cfg143	

Name	Description
gpio_cfg144	

4.8.1 gpio_cfg0

Address: 0x200008c4

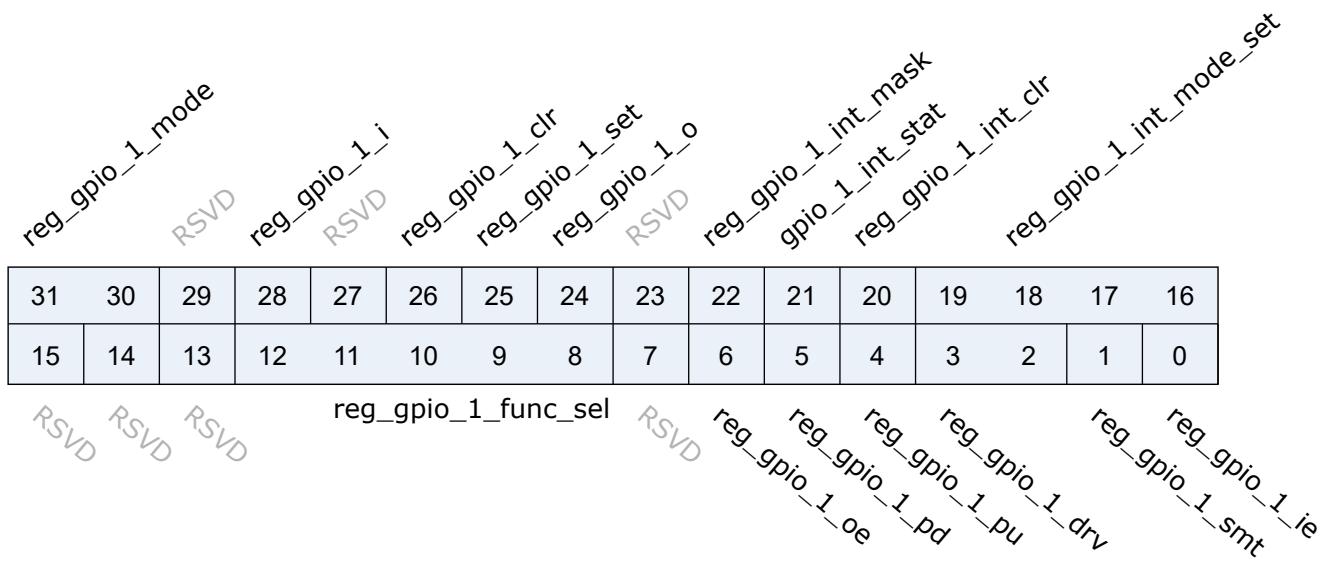


Bits	Name	Type	Reset	Description
31:30	reg_gpio_0_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_0_i	r	0	
27	RSVD			
26	reg_gpio_0_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
25	reg_gpio_0_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_0_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_0_int_mask	r/w	1	mask interrupt (1)
21	gpio_0_int_stat	r	0	interrupt status
20	reg_gpio_0_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_0_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_0_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_0_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_0_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_0_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_0_drv	r/w	0	GPIO Driving Control
1	reg_gpio_0_smt	r/w	1	GPIO SMT Control
0	reg_gpio_0_ie	r/w	0	GPIO Input Enable

4.8.2 gpio_cfg1

Address: 0x2000008c8

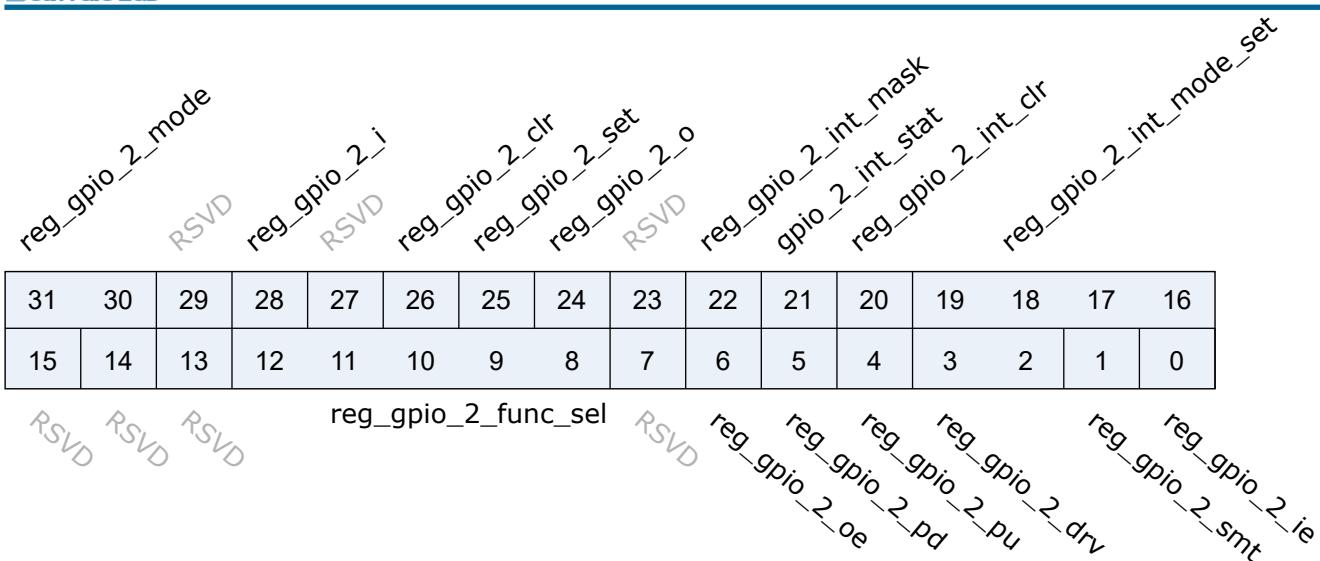


Bits	Name	Type	Reset	Description
31:30	reg_gpio_1_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_1_i	r	0	
27	RSVD			
26	reg_gpio_1_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_1_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_1_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_1_int_mask	r/w	1	mask interrupt (1)
21	gpio_1_int_stat	r	0	interrupt status
20	reg_gpio_1_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_1_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_1_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_1_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_1_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_1_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_1_drv	r/w	0	GPIO Driving Control
1	reg_gpio_1_smt	r/w	1	GPIO SMT Control
0	reg_gpio_1_ie	r/w	0	GPIO Input Enable

4.8.3 gpio_cfg2

Address: 0x2000008cc

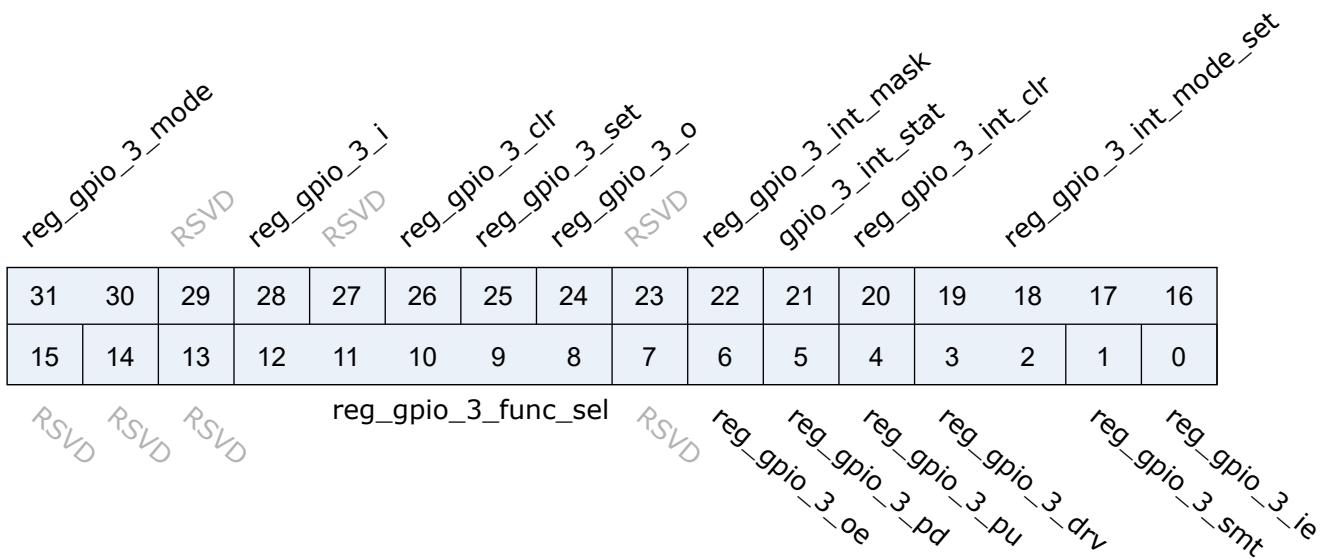


Bits	Name	Type	Reset	Description
31:30	reg_gpio_2_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_2_i	r	0	
27	RSVD			
26	reg_gpio_2_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_2_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_2_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_2_int_mask	r/w	1	mask interrupt (1)
21	gpio_2_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_2_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_2_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_2_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_2_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_2_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_2_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_2_drv	r/w	0	GPIO Driving Control
1	reg_gpio_2_smt	r/w	1	GPIO SMT Control
0	reg_gpio_2_ie	r/w	0	GPIO Input Enable

4.8.4 gpio_cfg3

Address: 0x200008d0

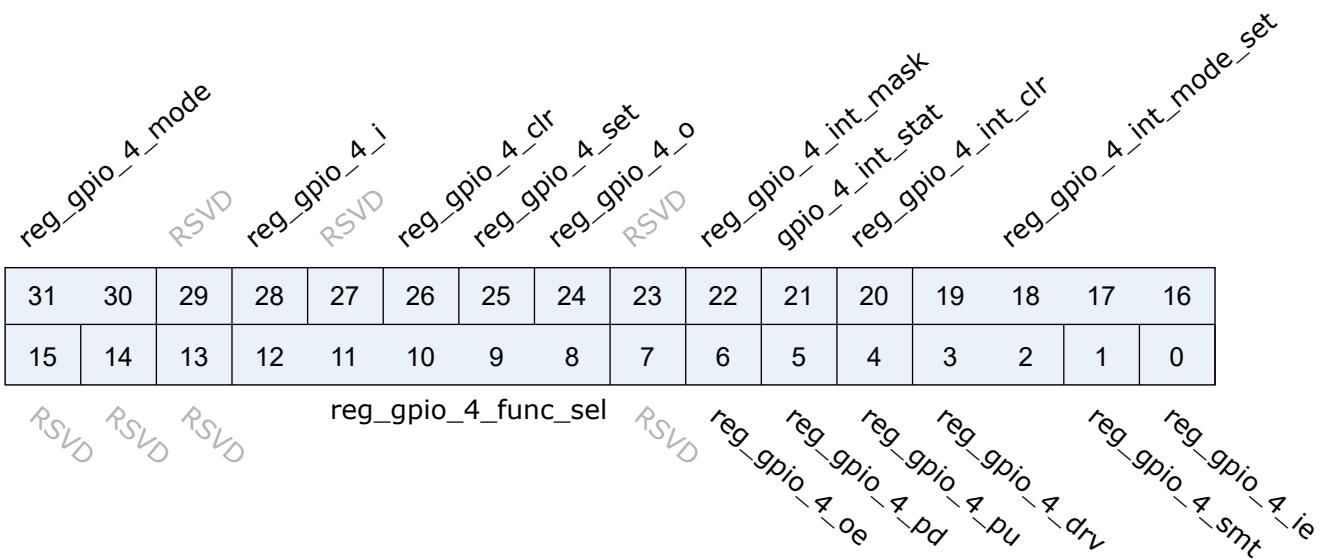


Bits	Name	Type	Reset	Description
31:30	reg_gpio_3_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_3_i	r	0	
27	RSVD			
26	reg_gpio_3_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_3_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_3_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_3_int_mask	r/w	1	mask interrupt (1)
21	gpio_3_int_stat	r	0	interrupt status
20	reg_gpio_3_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_3_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_3_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			

Bits	Name	Type	Reset	Description
6	reg_gpio_3_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_3_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_3_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_3_drv	r/w	0	GPIO Driving Control
1	reg_gpio_3_smt	r/w	1	GPIO SMT Control
0	reg_gpio_3_ie	r/w	0	GPIO Input Enable

4.8.5 gpio_cfg4

Address: 0x2000008d4

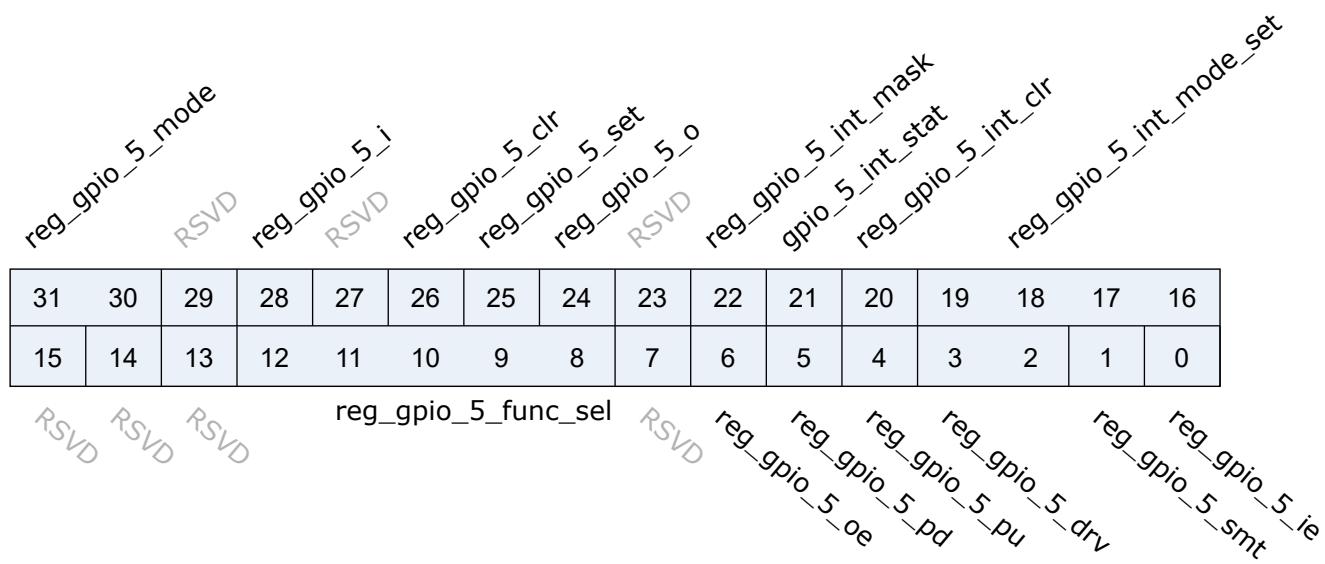


Bits	Name	Type	Reset	Description
31:30	reg_gpio_4_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			

Bits	Name	Type	Reset	Description
28	reg_gpio_4_i	r	0	
27	RSVD			
26	reg_gpio_4_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_4_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_4_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_4_int_mask	r/w	1	mask interrupt (1)
21	gpio_4_int_stat	r	0	interrupt status
20	reg_gpio_4_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_4_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_4_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_4_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_4_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_4_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_4_drv	r/w	0	GPIO Driving Control
1	reg_gpio_4_smt	r/w	1	GPIO SMT Control
0	reg_gpio_4_ie	r/w	0	GPIO Input Enable

4.8.6 gpio_cfg5

Address: 0x2000008d8

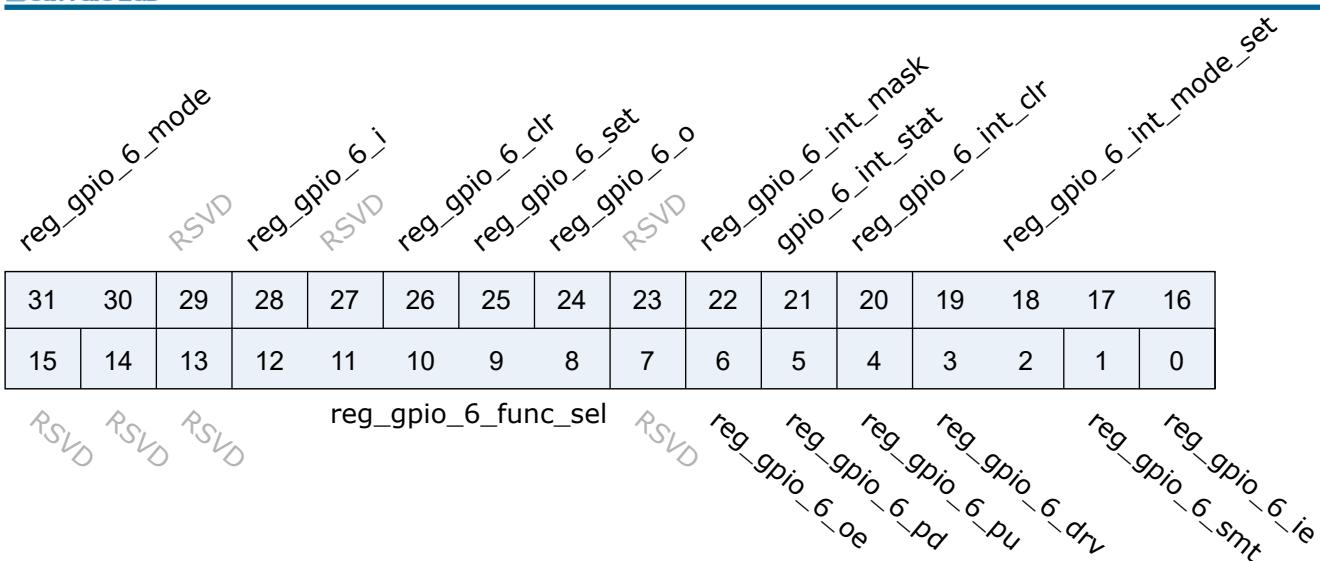


Bits	Name	Type	Reset	Description
31:30	reg_gpio_5_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_5_i	r	0	
27	RSVD			
26	reg_gpio_5_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_5_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_5_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_5_int_mask	r/w	1	mask interrupt (1)
21	gpio_5_int_stat	r	0	interrupt status
20	reg_gpio_5_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_5_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_5_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_5_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_5_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_5_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_5_drv	r/w	0	GPIO Driving Control
1	reg_gpio_5_smt	r/w	1	GPIO SMT Control
0	reg_gpio_5_ie	r/w	0	GPIO Input Enable

4.8.7 gpio_cfg6

Address: 0x2000008dc

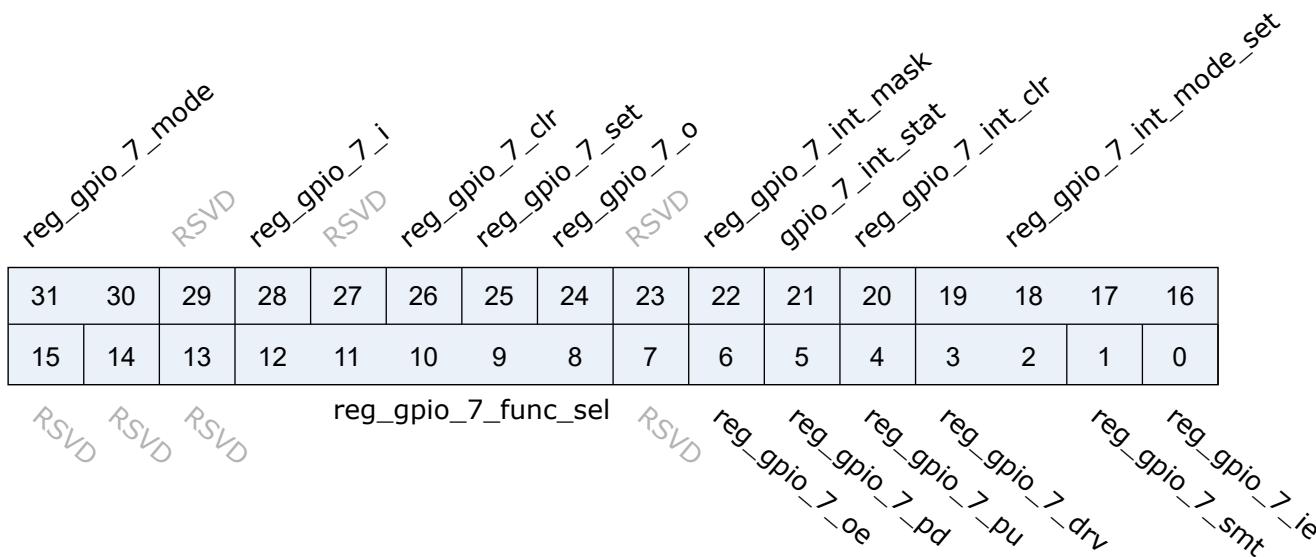


Bits	Name	Type	Reset	Description
31:30	reg_gpio_6_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_6_i	r	0	
27	RSVD			
26	reg_gpio_6_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_6_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_6_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_6_int_mask	r/w	1	mask interrupt (1)
21	gpio_6_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_6_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_6_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_6_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_6_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_6_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_6_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_6_drv	r/w	0	GPIO Driving Control
1	reg_gpio_6_smt	r/w	1	GPIO SMT Control
0	reg_gpio_6_ie	r/w	0	GPIO Input Enable

4.8.8 gpio_cfg7

Address: 0x2000008e0

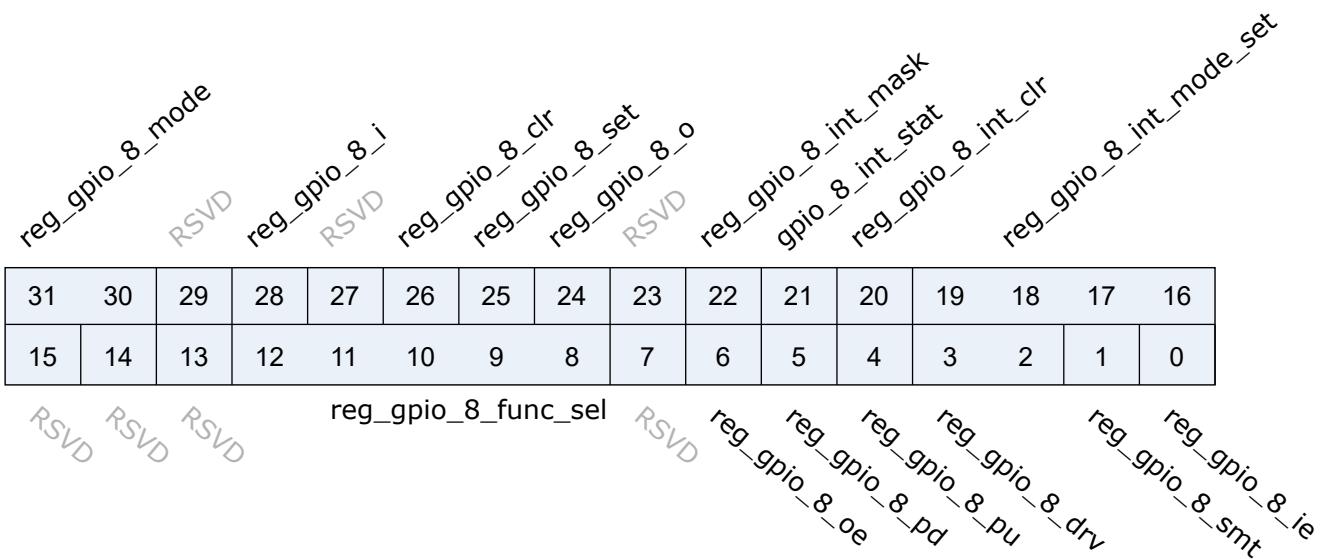


Bits	Name	Type	Reset	Description
31:30	reg_gpio_7_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_7_i	r	0	
27	RSVD			
26	reg_gpio_7_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_7_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_7_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_7_int_mask	r/w	1	mask interrupt (1)
21	gpio_7_int_stat	r	0	interrupt status
20	reg_gpio_7_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_7_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_7_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			

Bits	Name	Type	Reset	Description
6	reg_gpio_7_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_7_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_7_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_7_drv	r/w	0	GPIO Driving Control
1	reg_gpio_7_smt	r/w	1	GPIO SMT Control
0	reg_gpio_7_ie	r/w	0	GPIO Input Enable

4.8.9 gpio_cfg8

Address: 0x2000008e4

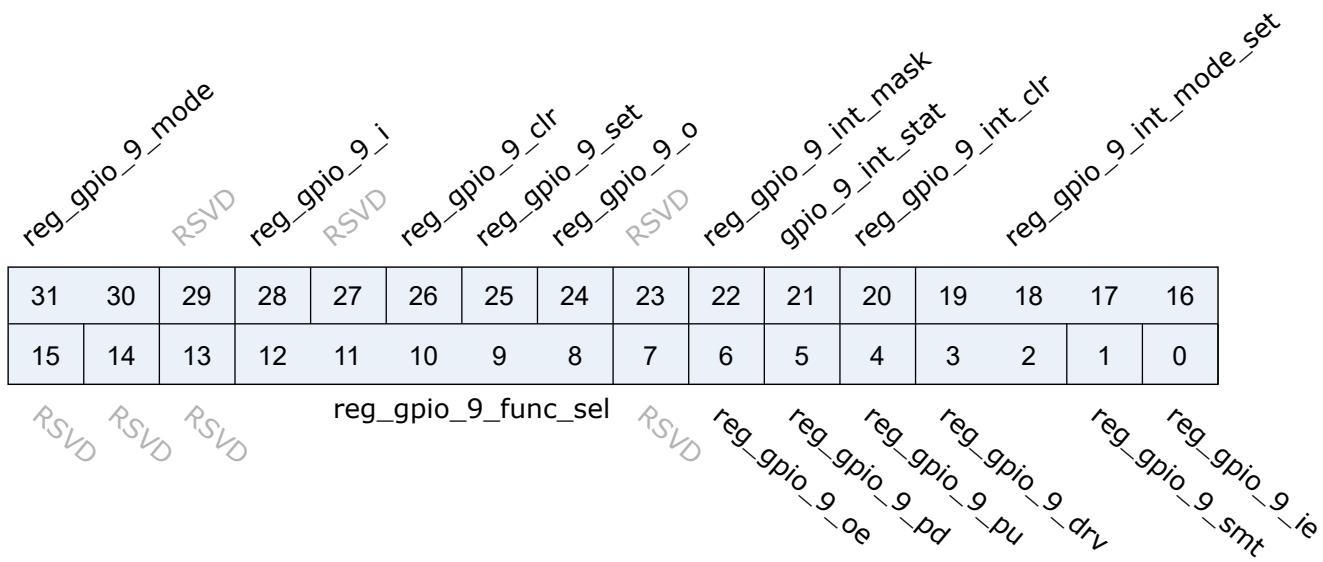


Bits	Name	Type	Reset	Description
31:30	reg_gpio_8_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			

Bits	Name	Type	Reset	Description
28	reg_gpio_8_i	r	0	
27	RSVD			
26	reg_gpio_8_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_8_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_8_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_8_int_mask	r/w	1	mask interrupt (1)
21	gpio_8_int_stat	r	0	interrupt status
20	reg_gpio_8_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_8_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_8_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_8_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_8_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_8_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_8_drv	r/w	0	GPIO Driving Control
1	reg_gpio_8_smt	r/w	1	GPIO SMT Control
0	reg_gpio_8_ie	r/w	0	GPIO Input Enable

4.8.10 gpio_cfg9

Address: 0x200008e8

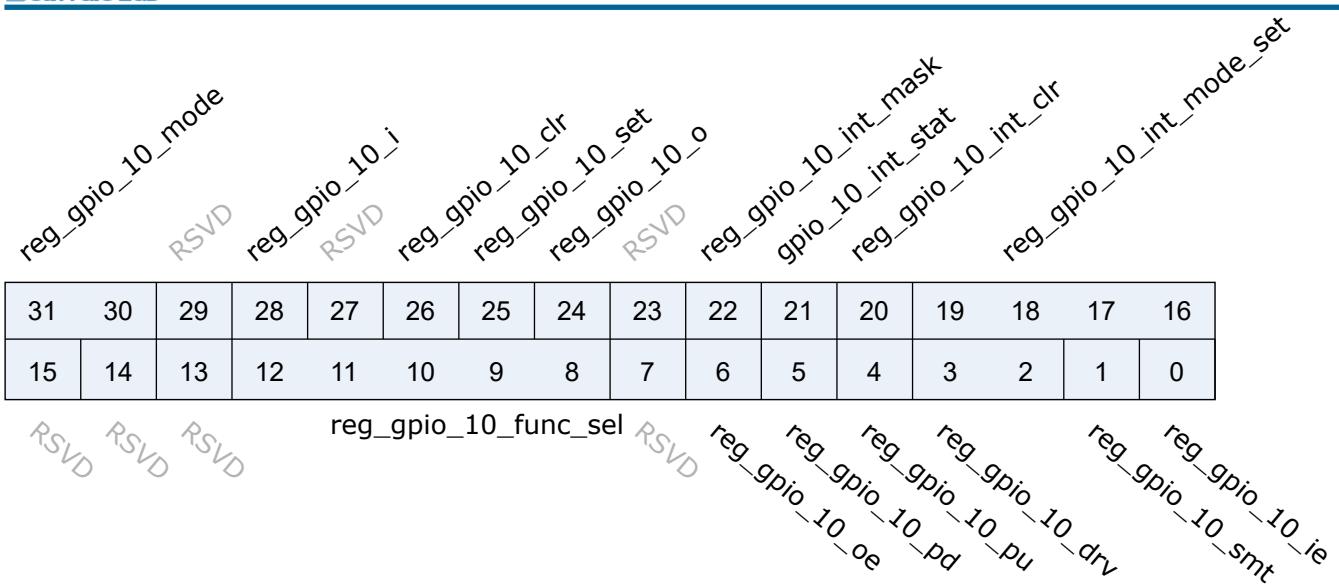


Bits	Name	Type	Reset	Description
31:30	reg_gpio_9_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_9_i	r	0	
27	RSVD			
26	reg_gpio_9_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_9_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_9_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_9_int_mask	r/w	1	mask interrupt (1)
21	gpio_9_int_stat	r	0	interrupt status
20	reg_gpio_9_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_9_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_9_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_9_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_9_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_9_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_9_drv	r/w	0	GPIO Driving Control
1	reg_gpio_9_smt	r/w	1	GPIO SMT Control
0	reg_gpio_9_ie	r/w	0	GPIO Input Enable

4.8.11 gpio_cfg10

Address: 0x2000008ec

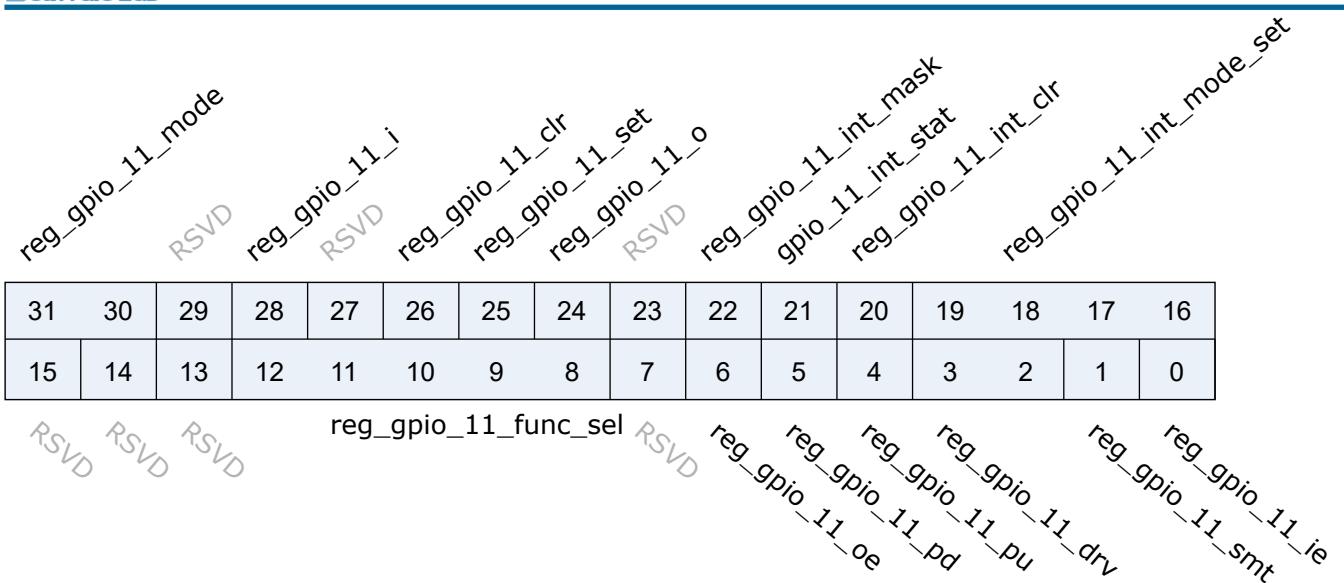


Bits	Name	Type	Reset	Description
31:30	reg_gpio_10_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_10_i	r	0	
27	RSVD			
26	reg_gpio_10_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_10_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_10_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_10_int_mask	r/w	1	mask interrupt (1)
21	gpio_10_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_10_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_10_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_10_func_sel	r/w	5'hF	GPIO Function Select (Default : CCI)
7	RSVD			
6	reg_gpio_10_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_10_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_10_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_10_drv	r/w	0	GPIO Driving Control
1	reg_gpio_10_smt	r/w	1	GPIO SMT Control
0	reg_gpio_10_ie	r/w	1	GPIO Input Enable

4.8.12 gpio_cfg11

Address: 0x2000008f0

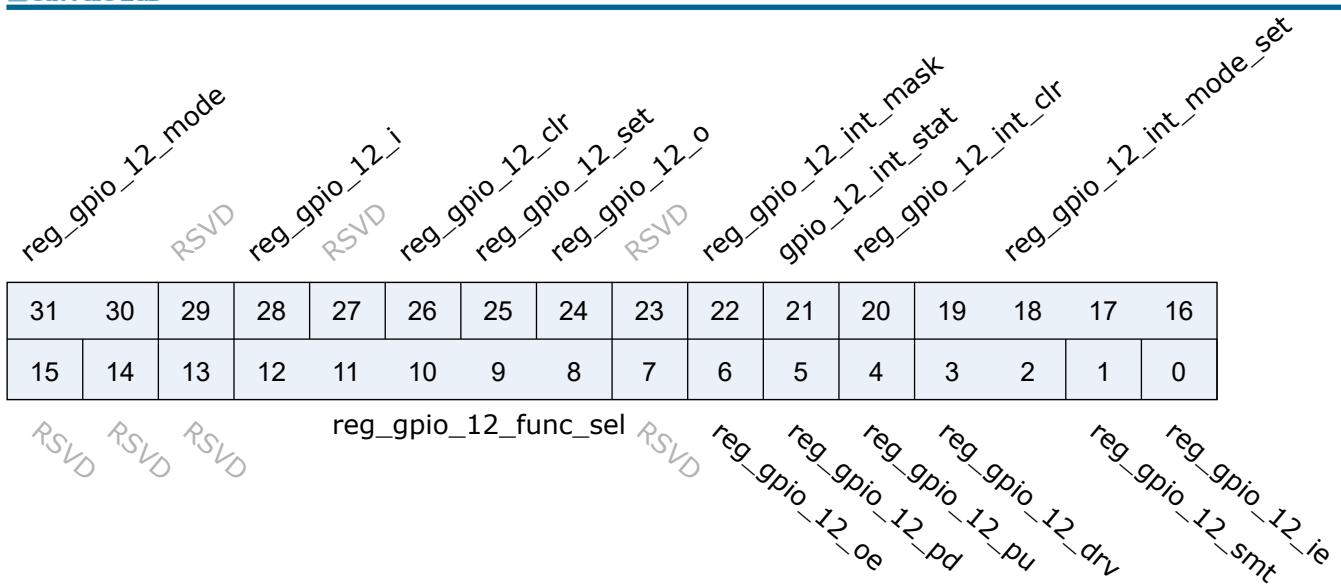


Bits	Name	Type	Reset	Description
31:30	reg_gpio_11_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_11_i	r	0	
27	RSVD			
26	reg_gpio_11_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_11_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_11_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_11_int_mask	r/w	1	mask interrupt (1)
21	gpio_11_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_11_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_11_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_11_func_sel	r/w	5'hF	GPIO Function Select (Default : CCI)
7	RSVD			
6	reg_gpio_11_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_11_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_11_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_11_drv	r/w	0	GPIO Driving Control
1	reg_gpio_11_smt	r/w	1	GPIO SMT Control
0	reg_gpio_11_ie	r/w	1	GPIO Input Enable

4.8.13 gpio_cfg12

Address: 0x2000008f4

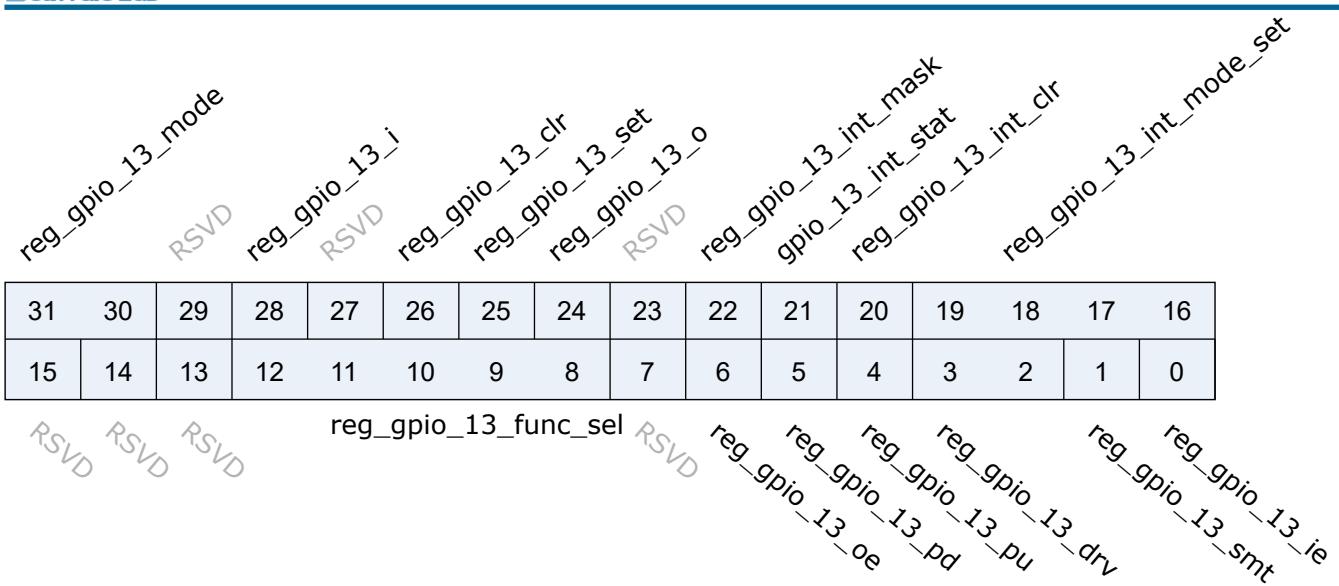


Bits	Name	Type	Reset	Description
31:30	reg_gpio_12_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_12_i	r	0	
27	RSVD			
26	reg_gpio_12_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_12_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_12_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_12_int_mask	r/w	1	mask interrupt (1)
21	gpio_12_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_12_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_12_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_12_func_sel	r/w	5'hF	GPIO Function Select (Default : CCI)
7	RSVD			
6	reg_gpio_12_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_12_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_12_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_12_drv	r/w	0	GPIO Driving Control
1	reg_gpio_12_smt	r/w	1	GPIO SMT Control
0	reg_gpio_12_ie	r/w	1	GPIO Input Enable

4.8.14 gpio_cfg13

Address: 0x2000008f8

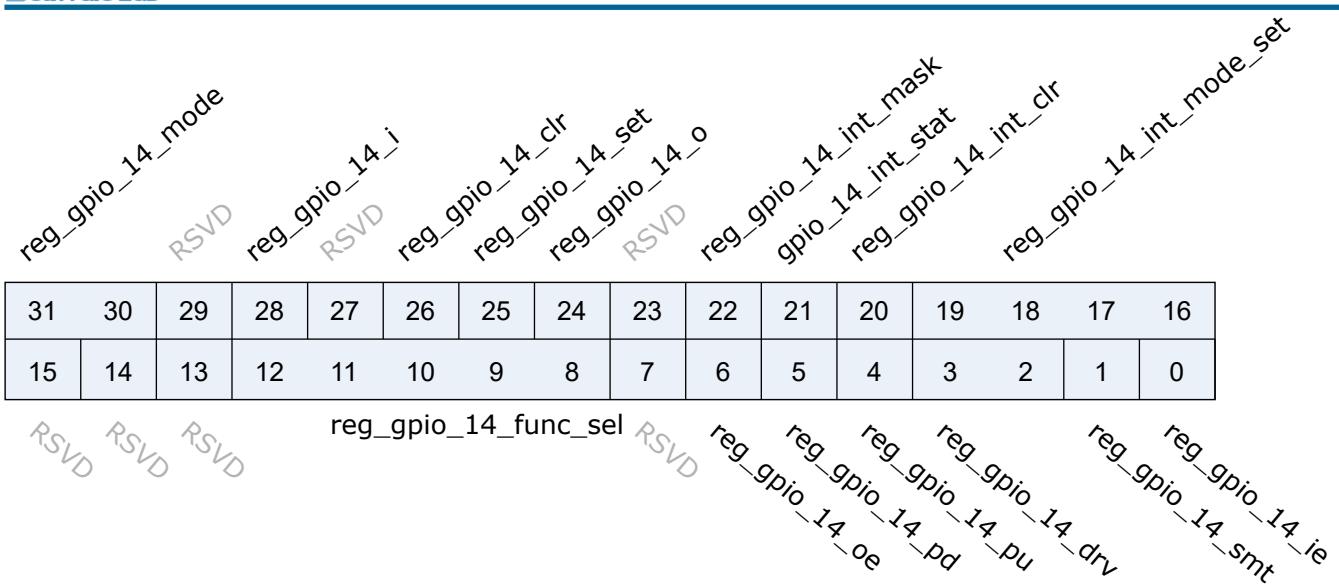


Bits	Name	Type	Reset	Description
31:30	reg_gpio_13_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_13_i	r	0	
27	RSVD			
26	reg_gpio_13_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_13_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_13_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_13_int_mask	r/w	1	mask interrupt (1)
21	gpio_13_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_13_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_13_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_13_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_13_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_13_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_13_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_13_drv	r/w	0	GPIO Driving Control
1	reg_gpio_13_smt	r/w	1	GPIO SMT Control
0	reg_gpio_13_ie	r/w	0	GPIO Input Enable

4.8.15 gpio_cfg14

Address: 0x2000008fc

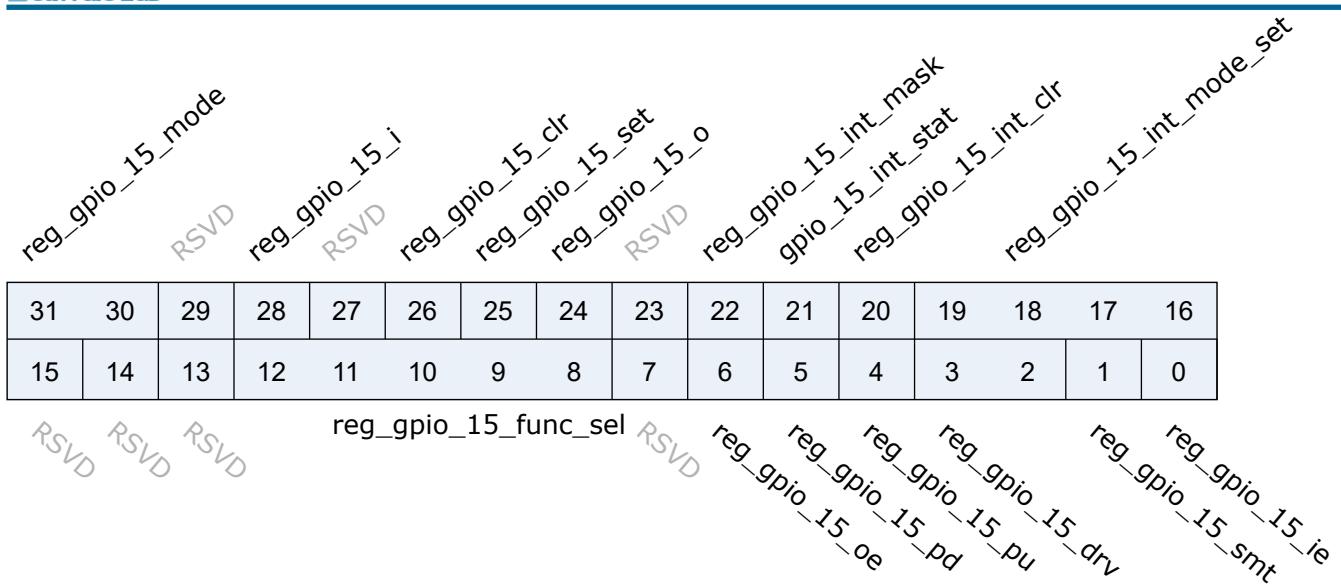


Bits	Name	Type	Reset	Description
31:30	reg_gpio_14_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_14_i	r	0	
27	RSVD			
26	reg_gpio_14_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_14_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_14_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_14_int_mask	r/w	1	mask interrupt (1)
21	gpio_14_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_14_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_14_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_14_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_14_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_14_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_14_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_14_drv	r/w	0	GPIO Driving Control
1	reg_gpio_14_smt	r/w	1	GPIO SMT Control
0	reg_gpio_14_ie	r/w	0	GPIO Input Enable

4.8.16 gpio_cfg15

Address: 0x20000900

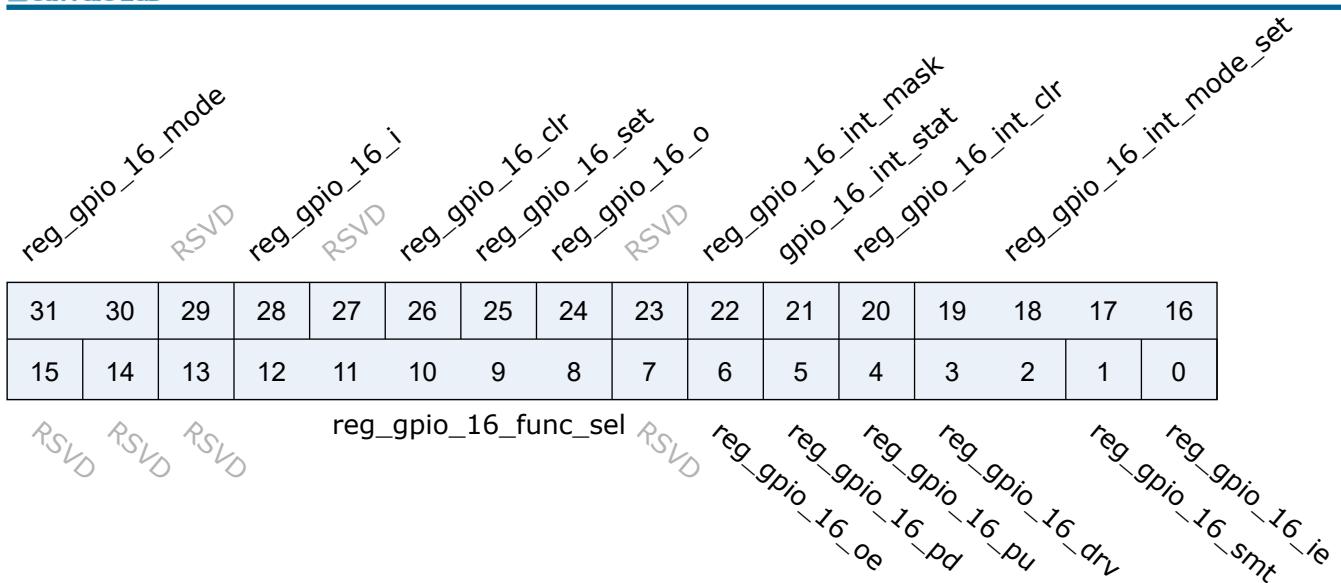


Bits	Name	Type	Reset	Description
31:30	reg_gpio_15_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_15_i	r	0	
27	RSVD			
26	reg_gpio_15_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_15_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_15_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_15_int_mask	r/w	1	mask interrupt (1)
21	gpio_15_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_15_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_15_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_15_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_15_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_15_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_15_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_15_drv	r/w	0	GPIO Driving Control
1	reg_gpio_15_smt	r/w	1	GPIO SMT Control
0	reg_gpio_15_ie	r/w	0	GPIO Input Enable

4.8.17 gpio_cfg16

Address: 0x20000904

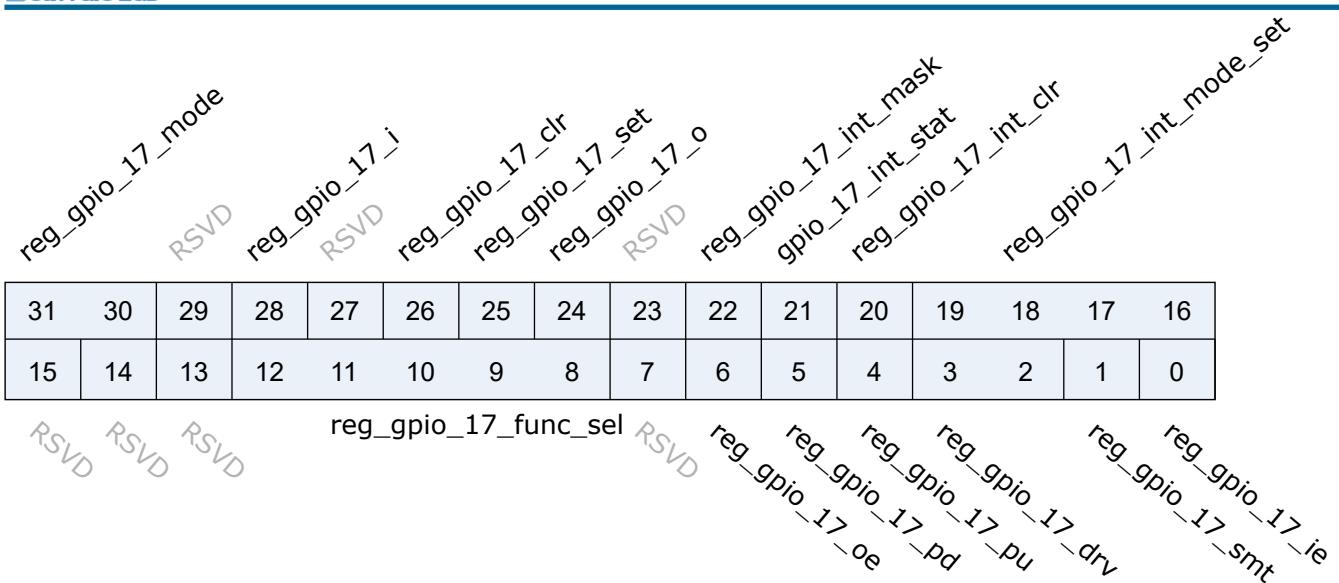


Bits	Name	Type	Reset	Description
31:30	reg_gpio_16_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_16_i	r	0	
27	RSVD			
26	reg_gpio_16_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_16_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_16_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_16_int_mask	r/w	1	mask interrupt (1)
21	gpio_16_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_16_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_16_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_16_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_16_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_16_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_16_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_16_drv	r/w	0	GPIO Driving Control
1	reg_gpio_16_smt	r/w	1	GPIO SMT Control
0	reg_gpio_16_ie	r/w	0	GPIO Input Enable

4.8.18 gpio_cfg17

Address: 0x20000908

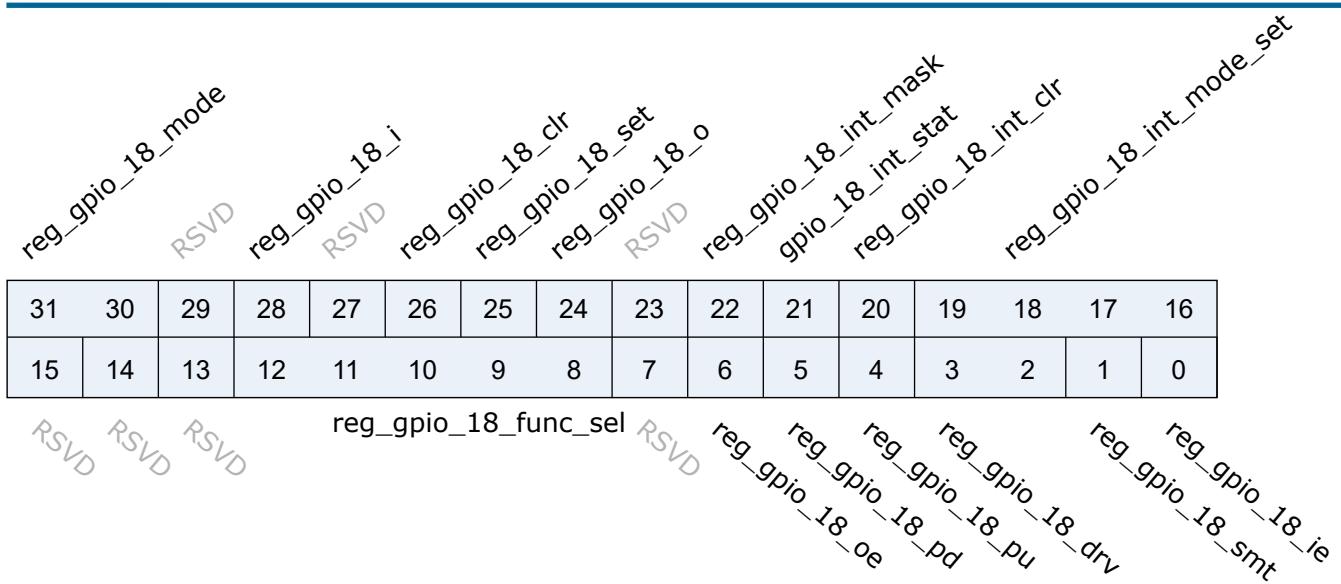


Bits	Name	Type	Reset	Description
31:30	reg_gpio_17_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_17_i	r	0	
27	RSVD			
26	reg_gpio_17_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_17_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_17_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_17_int_mask	r/w	1	mask interrupt (1)
21	gpio_17_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_17_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_17_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_17_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_17_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_17_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_17_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_17_drv	r/w	0	GPIO Driving Control
1	reg_gpio_17_smt	r/w	1	GPIO SMT Control
0	reg_gpio_17_ie	r/w	0	GPIO Input Enable

4.8.19 gpio_cfg18

Address: 0x2000090c

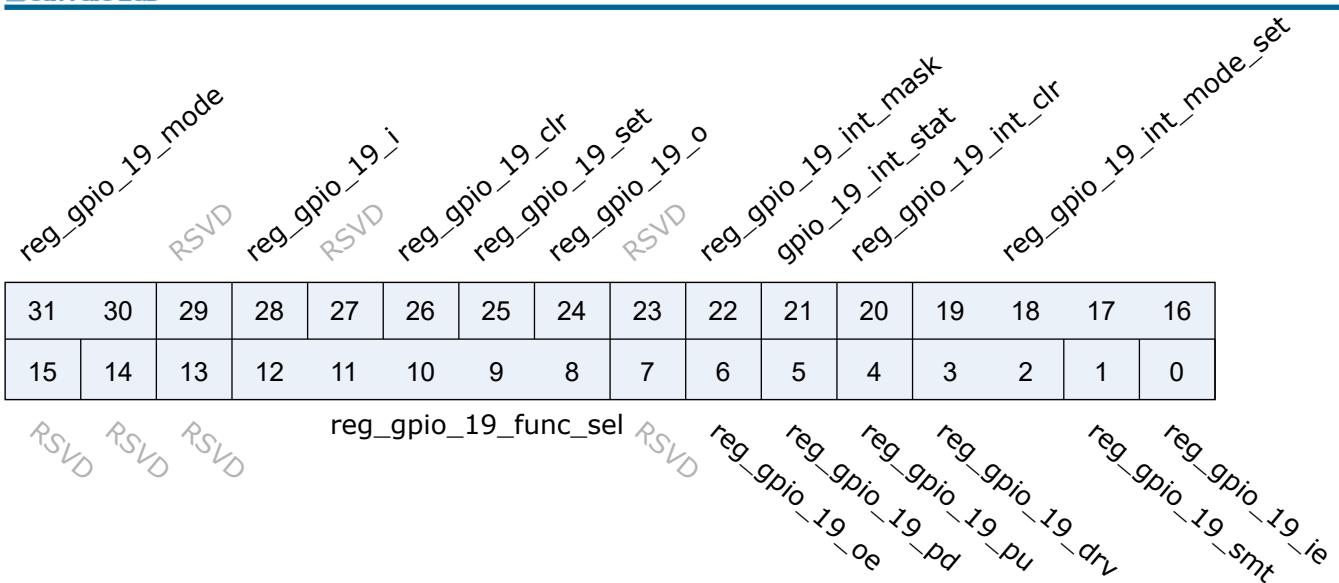


Bits	Name	Type	Reset	Description
31:30	reg_gpio_18_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_18_i	r	0	
27	RSVD			
26	reg_gpio_18_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_18_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_18_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_18_int_mask	r/w	1	mask interrupt (1)
21	gpio_18_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_18_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_18_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_18_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_18_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_18_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_18_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_18_drv	r/w	0	GPIO Driving Control
1	reg_gpio_18_smt	r/w	1	GPIO SMT Control
0	reg_gpio_18_ie	r/w	0	GPIO Input Enable

4.8.20 gpio_cfg19

Address: 0x20000910

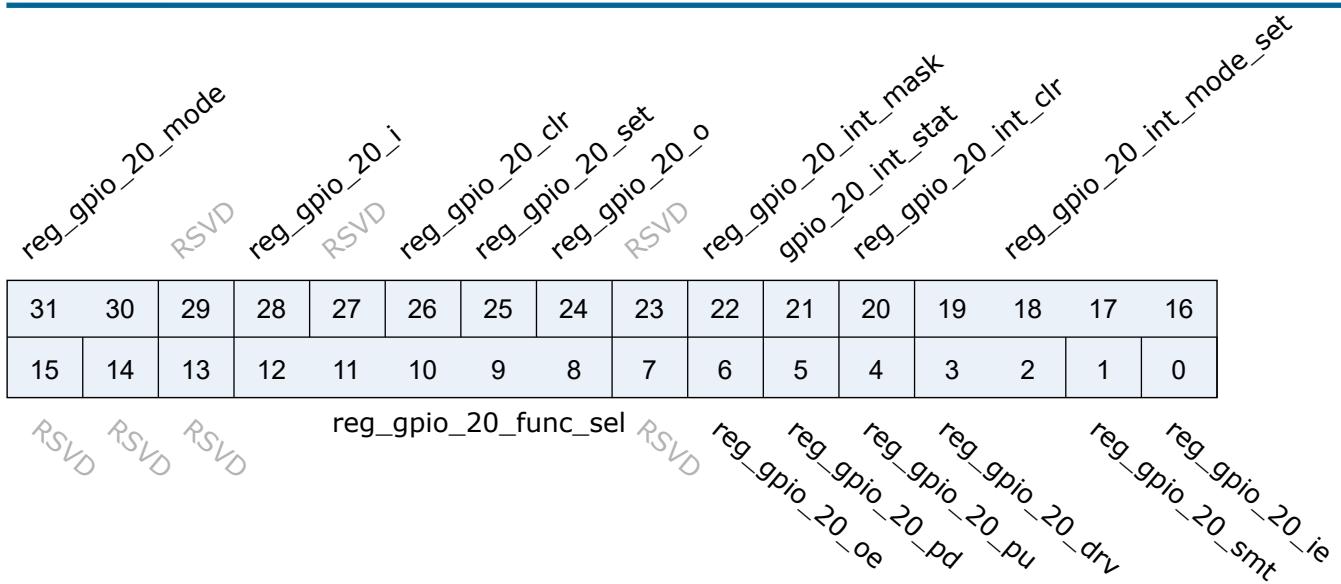


Bits	Name	Type	Reset	Description
31:30	reg_gpio_19_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_19_i	r	0	
27	RSVD			
26	reg_gpio_19_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_19_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_19_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_19_int_mask	r/w	1	mask interrupt (1)
21	gpio_19_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_19_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_19_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_19_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_19_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_19_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_19_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_19_drv	r/w	0	GPIO Driving Control
1	reg_gpio_19_smt	r/w	1	GPIO SMT Control
0	reg_gpio_19_ie	r/w	0	GPIO Input Enable

4.8.21 gpio_cfg20

Address: 0x20000914

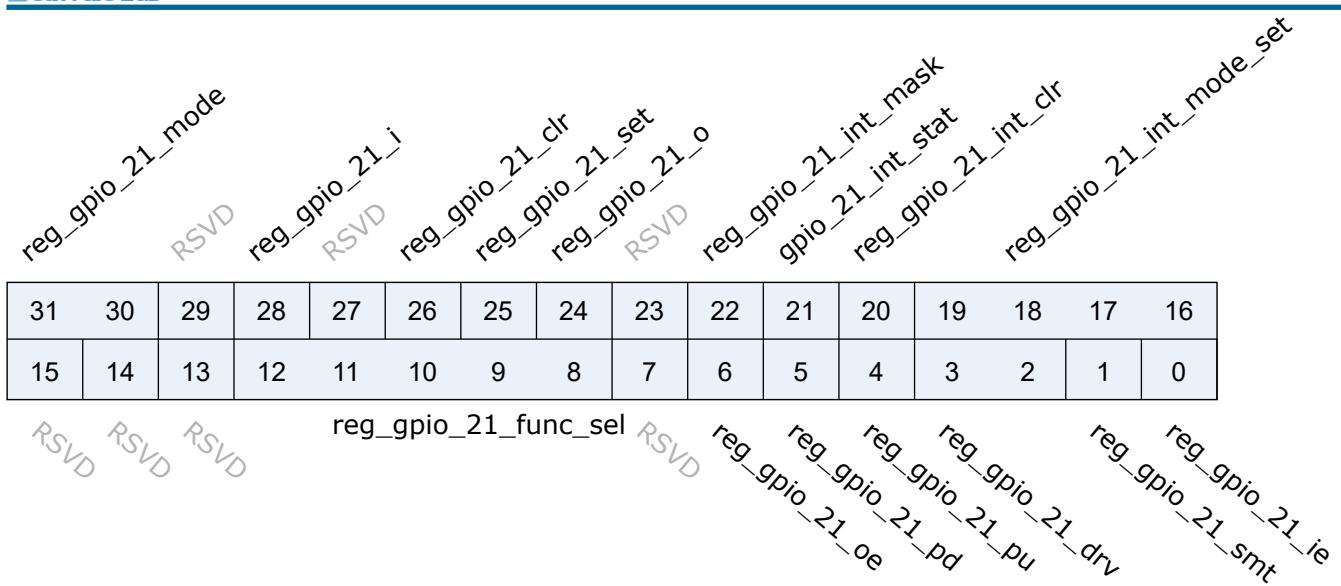


Bits	Name	Type	Reset	Description
31:30	reg_gpio_20_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_20_i	r	0	
27	RSVD			
26	reg_gpio_20_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_20_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_20_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_20_int_mask	r/w	1	mask interrupt (1)
21	gpio_20_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_20_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_20_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_20_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_20_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_20_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_20_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_20_drv	r/w	0	GPIO Driving Control
1	reg_gpio_20_smt	r/w	1	GPIO SMT Control
0	reg_gpio_20_ie	r/w	0	GPIO Input Enable

4.8.22 gpio_cfg21

Address: 0x20000918

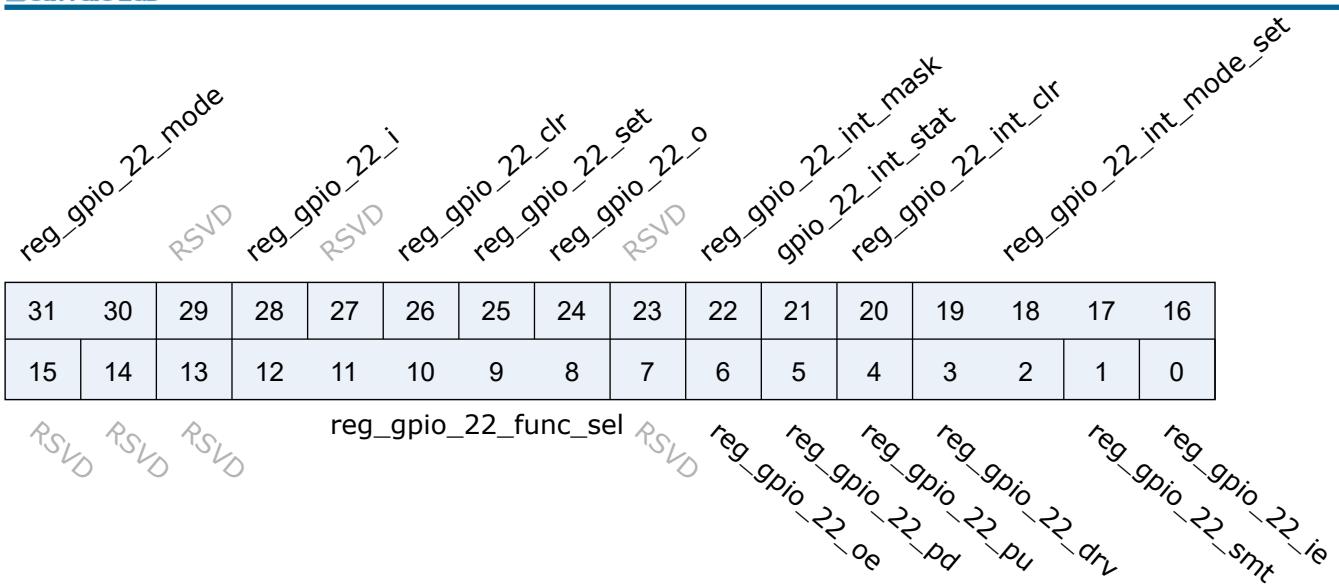


Bits	Name	Type	Reset	Description
31:30	reg_gpio_21_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_21_i	r	0	
27	RSVD			
26	reg_gpio_21_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_21_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_21_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_21_int_mask	r/w	1	mask interrupt (1)
21	gpio_21_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_21_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_21_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_21_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_21_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_21_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_21_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_21_drv	r/w	0	GPIO Driving Control
1	reg_gpio_21_smt	r/w	1	GPIO SMT Control
0	reg_gpio_21_ie	r/w	0	GPIO Input Enable

4.8.23 gpio_cfg22

Address: 0x2000091c

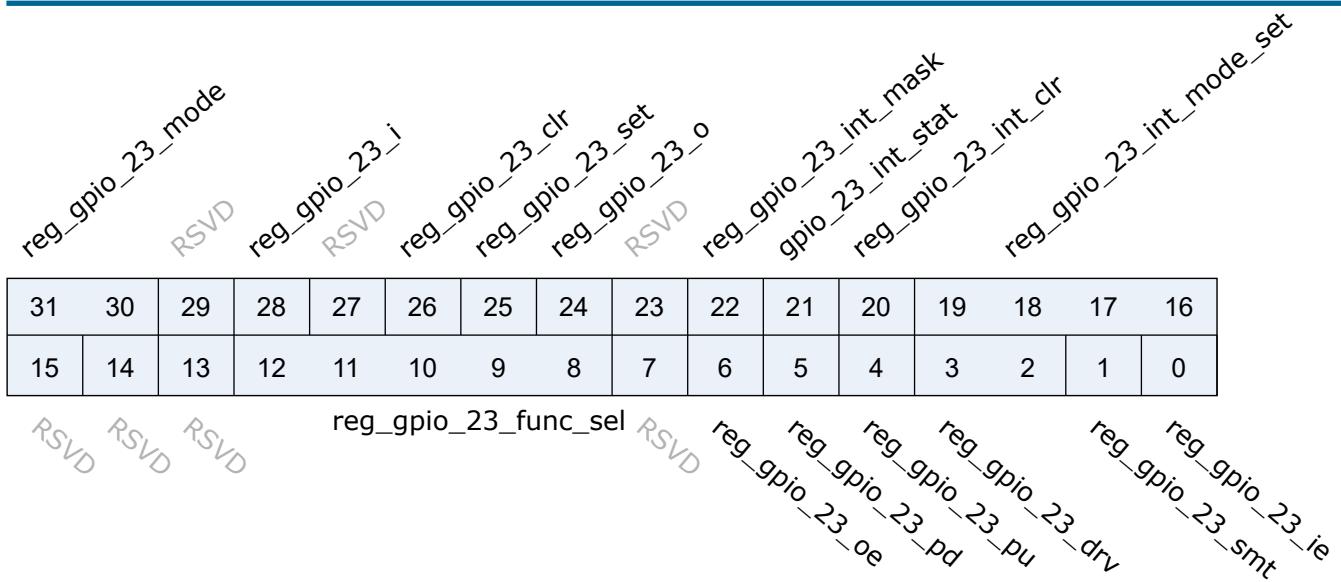


Bits	Name	Type	Reset	Description
31:30	reg_gpio_22_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_22_i	r	0	
27	RSVD			
26	reg_gpio_22_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_22_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_22_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_22_int_mask	r/w	1	mask interrupt (1)
21	gpio_22_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_22_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_22_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_22_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_22_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_22_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_22_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_22_drv	r/w	0	GPIO Driving Control
1	reg_gpio_22_smt	r/w	1	GPIO SMT Control
0	reg_gpio_22_ie	r/w	0	GPIO Input Enable

4.8.24 gpio_cfg23

Address: 0x20000920

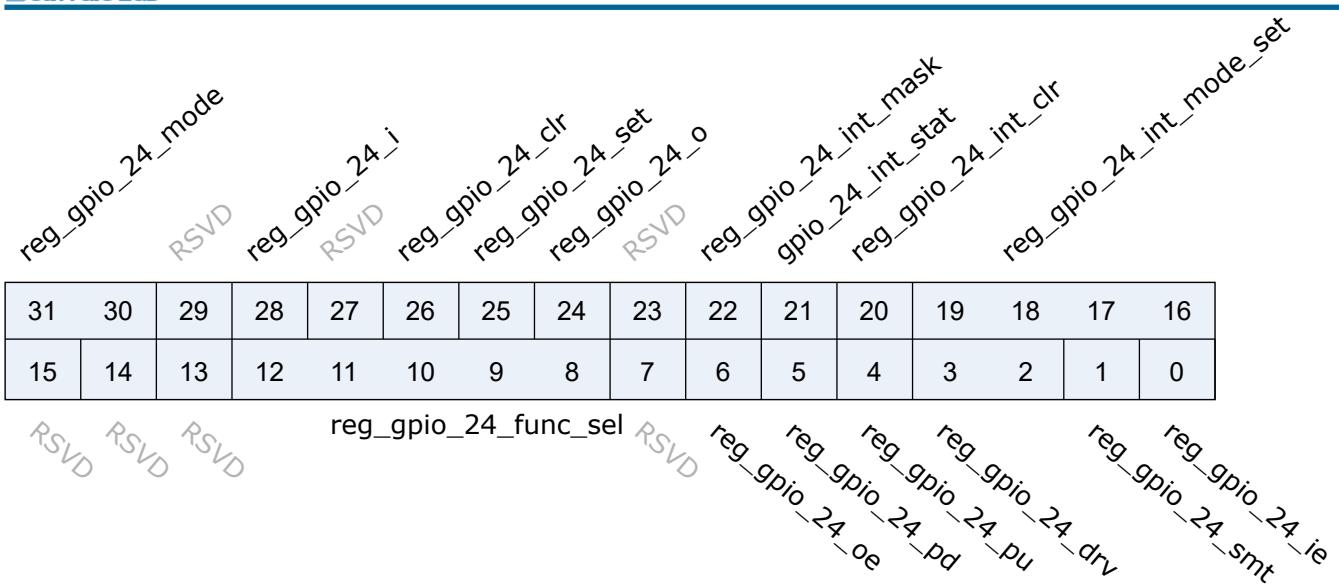


Bits	Name	Type	Reset	Description
31:30	reg_gpio_23_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_23_i	r	0	
27	RSVD			
26	reg_gpio_23_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_23_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_23_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_23_int_mask	r/w	1	mask interrupt (1)
21	gpio_23_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_23_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_23_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_23_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_23_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_23_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_23_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_23_drv	r/w	0	GPIO Driving Control
1	reg_gpio_23_smt	r/w	1	GPIO SMT Control
0	reg_gpio_23_ie	r/w	0	GPIO Input Enable

4.8.25 gpio_cfg24

Address: 0x20000924

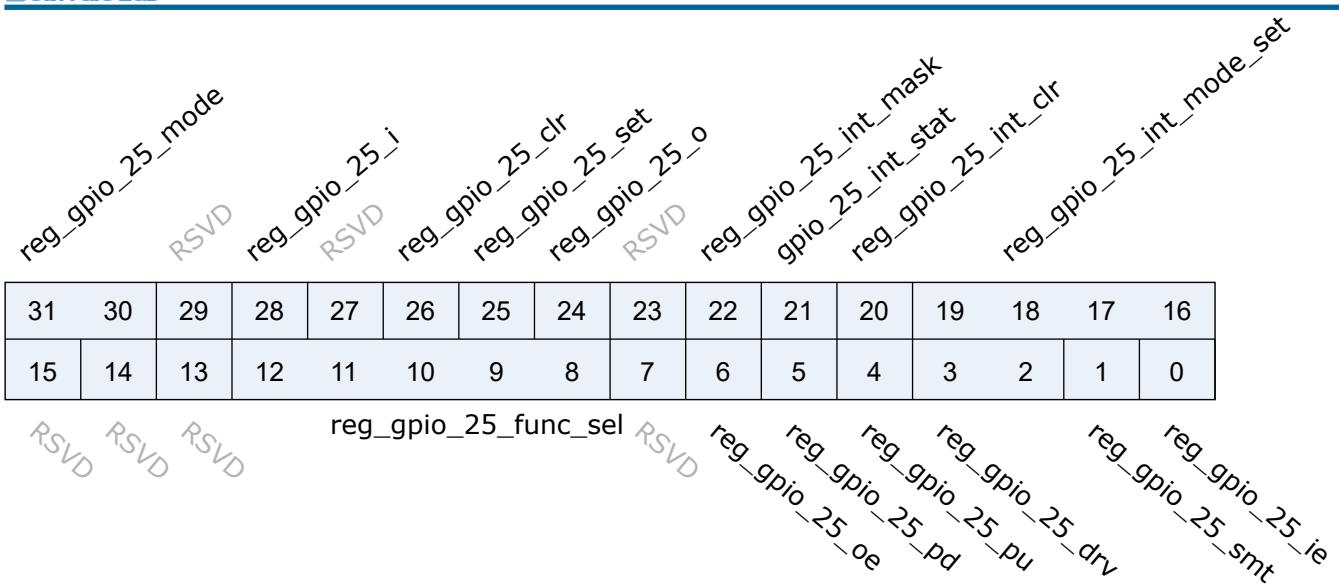


Bits	Name	Type	Reset	Description
31:30	reg_gpio_24_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_24_i	r	0	
27	RSVD			
26	reg_gpio_24_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_24_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_24_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_24_int_mask	r/w	1	mask interrupt (1)
21	gpio_24_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_24_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_24_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_24_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_24_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_24_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_24_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_24_drv	r/w	0	GPIO Driving Control
1	reg_gpio_24_smt	r/w	1	GPIO SMT Control
0	reg_gpio_24_ie	r/w	0	GPIO Input Enable

4.8.26 gpio_cfg25

Address: 0x20000928

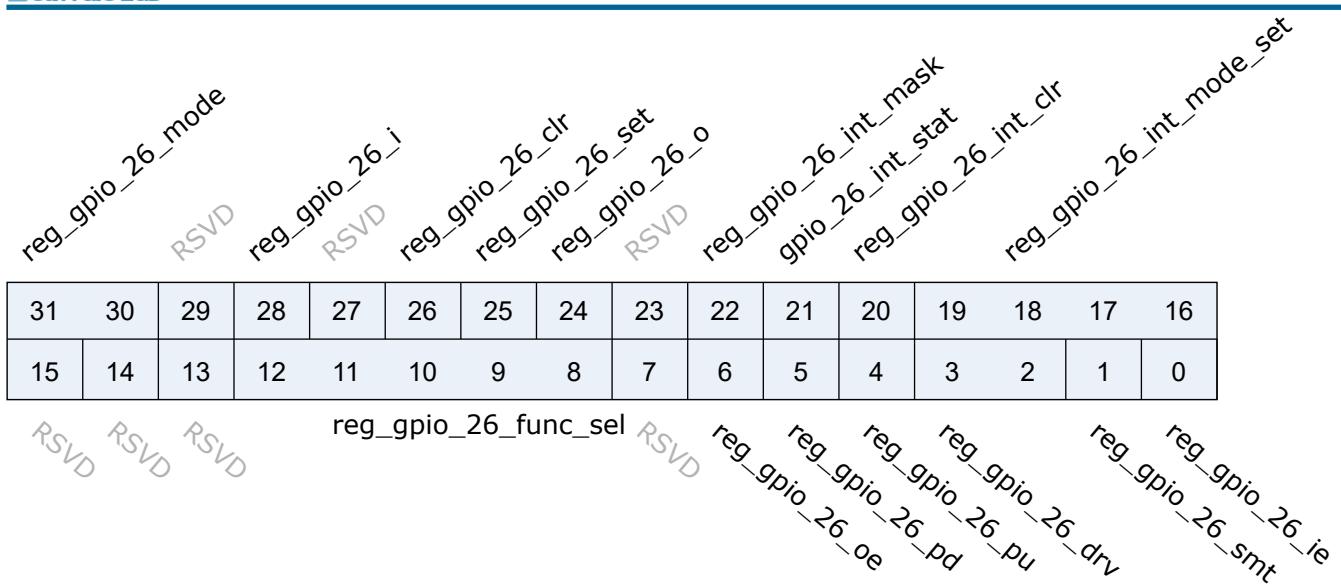


Bits	Name	Type	Reset	Description
31:30	reg_gpio_25_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_25_i	r	0	
27	RSVD			
26	reg_gpio_25_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_25_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_25_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_25_int_mask	r/w	1	mask interrupt (1)
21	gpio_25_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_25_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_25_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_25_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_25_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_25_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_25_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_25_drv	r/w	0	GPIO Driving Control
1	reg_gpio_25_smt	r/w	1	GPIO SMT Control
0	reg_gpio_25_ie	r/w	0	GPIO Input Enable

4.8.27 gpio_cfg26

Address: 0x2000092c

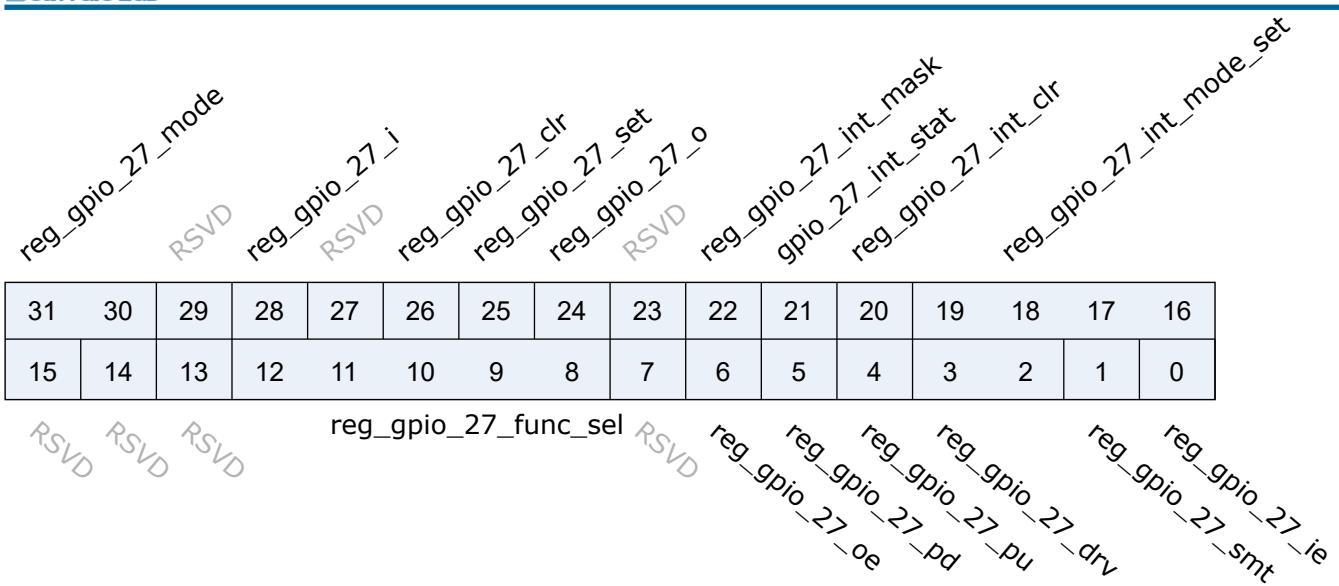


Bits	Name	Type	Reset	Description
31:30	reg_gpio_26_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_26_i	r	0	
27	RSVD			
26	reg_gpio_26_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_26_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_26_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_26_int_mask	r/w	1	mask interrupt (1)
21	gpio_26_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_26_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_26_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_26_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_26_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_26_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_26_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_26_drv	r/w	0	GPIO Driving Control
1	reg_gpio_26_smt	r/w	1	GPIO SMT Control
0	reg_gpio_26_ie	r/w	0	GPIO Input Enable

4.8.28 gpio_cfg27

Address: 0x20000930

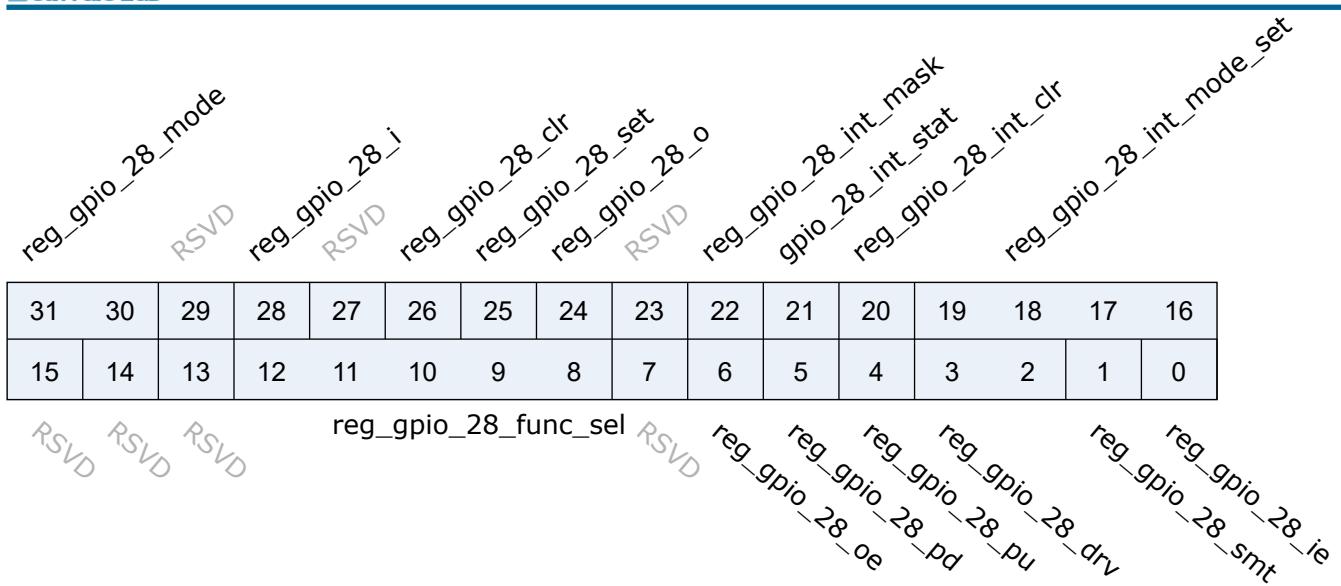


Bits	Name	Type	Reset	Description
31:30	reg_gpio_27_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_27_i	r	0	
27	RSVD			
26	reg_gpio_27_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_27_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_27_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_27_int_mask	r/w	1	mask interrupt (1)
21	gpio_27_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_27_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_27_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_27_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_27_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_27_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_27_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_27_drv	r/w	0	GPIO Driving Control
1	reg_gpio_27_smt	r/w	1	GPIO SMT Control
0	reg_gpio_27_ie	r/w	0	GPIO Input Enable

4.8.29 gpio_cfg28

Address: 0x20000934

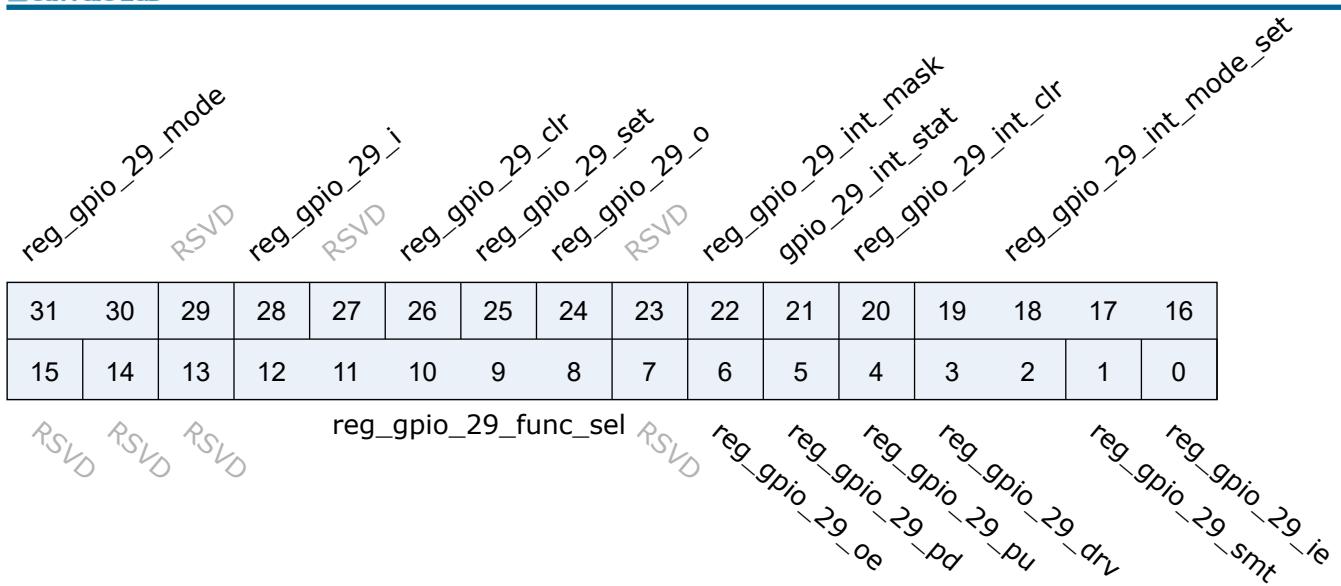


Bits	Name	Type	Reset	Description
31:30	reg_gpio_28_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_28_i	r	0	
27	RSVD			
26	reg_gpio_28_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_28_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_28_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_28_int_mask	r/w	1	mask interrupt (1)
21	gpio_28_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_28_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_28_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_28_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_28_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_28_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_28_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_28_drv	r/w	0	GPIO Driving Control
1	reg_gpio_28_smt	r/w	1	GPIO SMT Control
0	reg_gpio_28_ie	r/w	0	GPIO Input Enable

4.8.30 gpio_cfg29

Address: 0x20000938

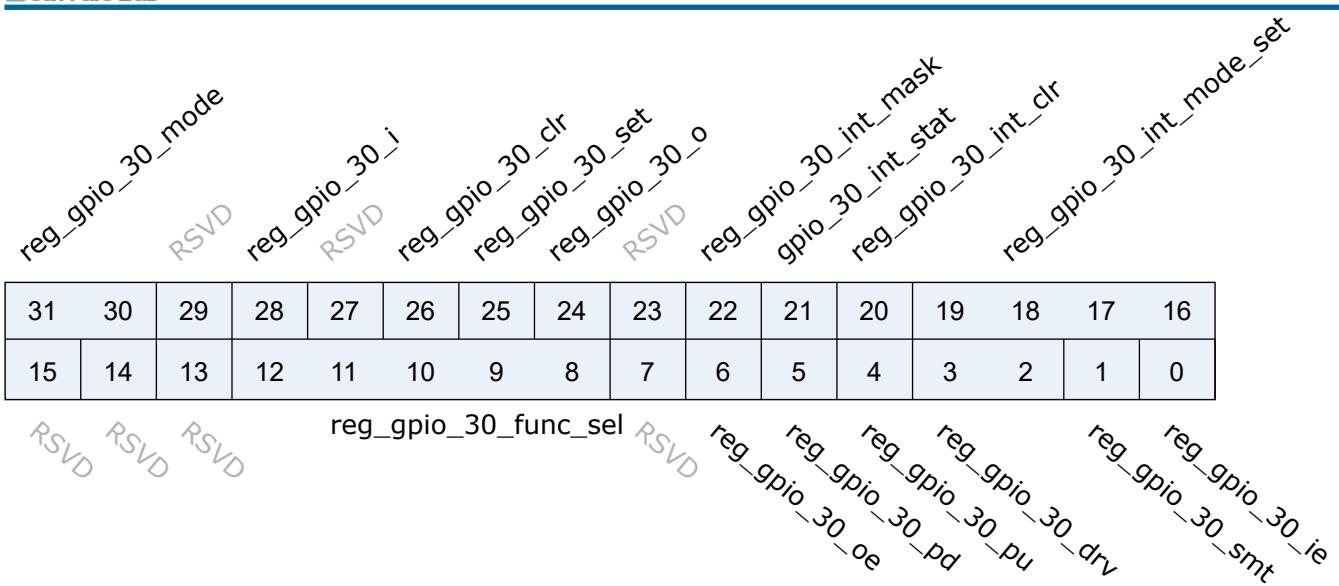


Bits	Name	Type	Reset	Description
31:30	reg_gpio_29_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_29_i	r	0	
27	RSVD			
26	reg_gpio_29_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_29_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_29_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_29_int_mask	r/w	1	mask interrupt (1)
21	gpio_29_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_29_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_29_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_29_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_29_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_29_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_29_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_29_drv	r/w	0	GPIO Driving Control
1	reg_gpio_29_smt	r/w	1	GPIO SMT Control
0	reg_gpio_29_ie	r/w	0	GPIO Input Enable

4.8.31 gpio_cfg30

Address: 0x2000093c

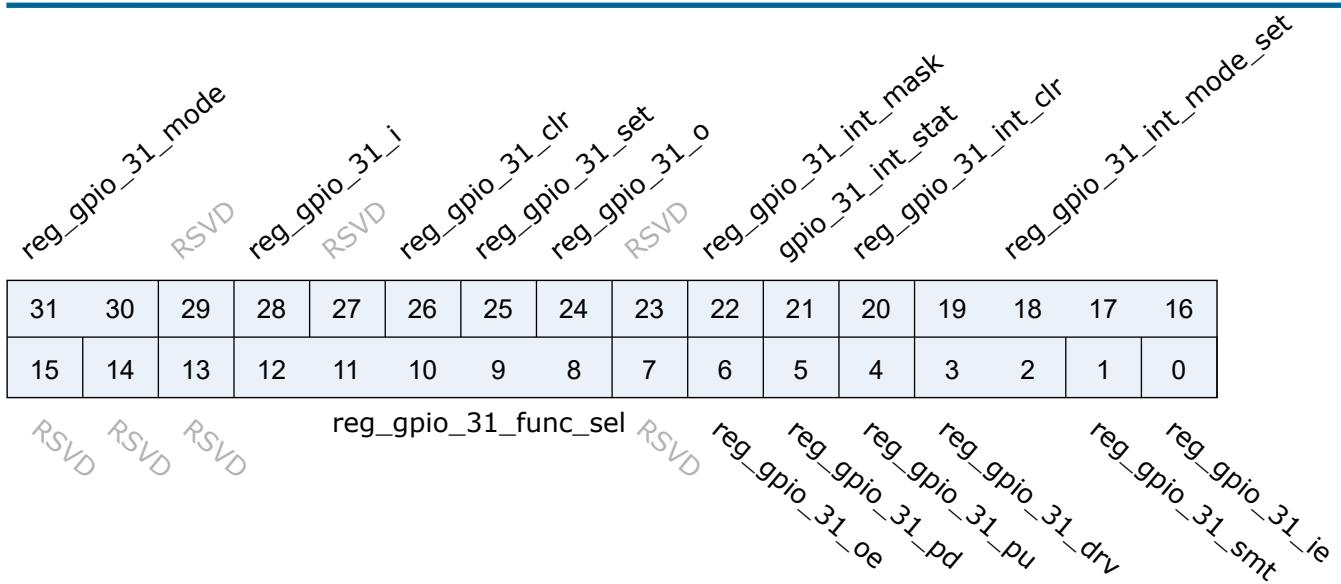


Bits	Name	Type	Reset	Description
31:30	reg_gpio_30_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_30_i	r	0	
27	RSVD			
26	reg_gpio_30_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_30_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_30_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_30_int_mask	r/w	1	mask interrupt (1)
21	gpio_30_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_30_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_30_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_30_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_30_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_30_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_30_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_30_drv	r/w	0	GPIO Driving Control
1	reg_gpio_30_smt	r/w	1	GPIO SMT Control
0	reg_gpio_30_ie	r/w	0	GPIO Input Enable

4.8.32 gpio_cfg31

Address: 0x20000940

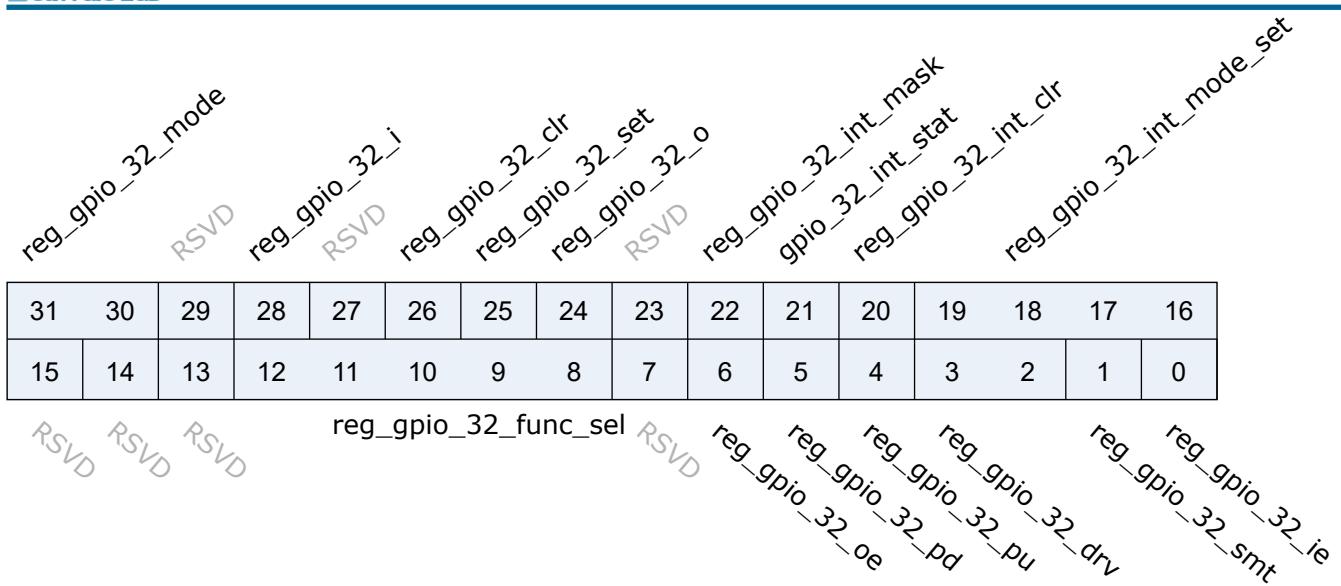


Bits	Name	Type	Reset	Description
31:30	reg_gpio_31_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_31_i	r	0	
27	RSVD			
26	reg_gpio_31_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_31_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_31_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_31_int_mask	r/w	1	mask interrupt (1)
21	gpio_31_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_31_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_31_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_31_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_31_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_31_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_31_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_31_drv	r/w	0	GPIO Driving Control
1	reg_gpio_31_smt	r/w	1	GPIO SMT Control
0	reg_gpio_31_ie	r/w	0	GPIO Input Enable

4.8.33 gpio_cfg32

Address: 0x20000944

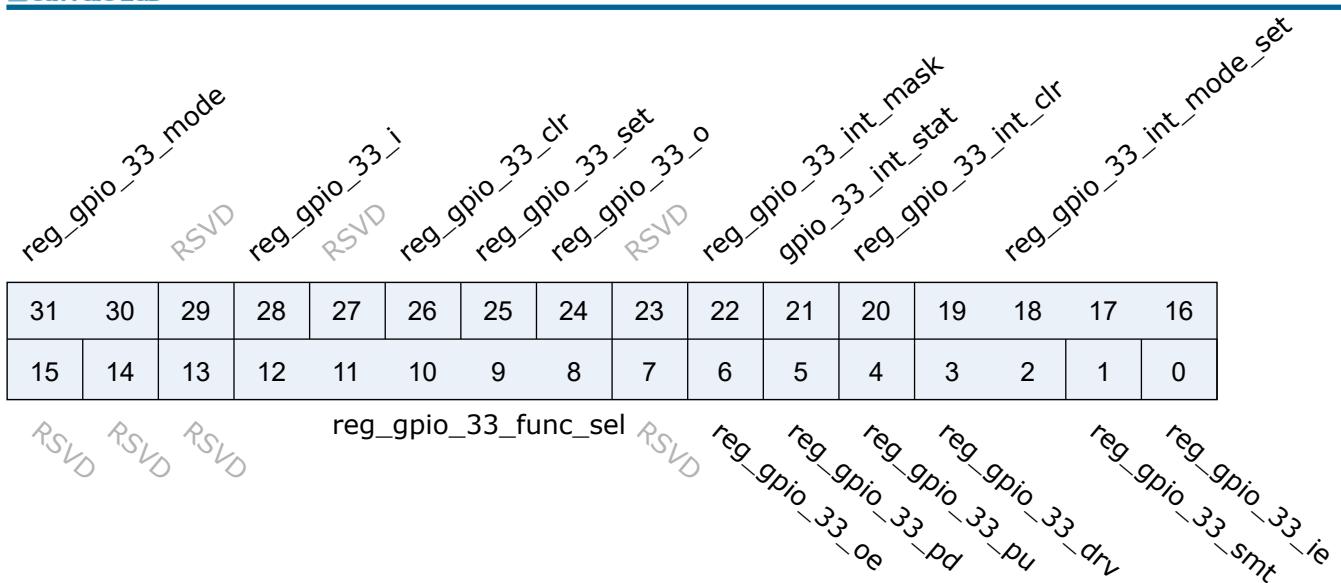


Bits	Name	Type	Reset	Description
31:30	reg_gpio_32_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_32_i	r	0	
27	RSVD			
26	reg_gpio_32_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_32_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_32_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_32_int_mask	r/w	1	mask interrupt (1)
21	gpio_32_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_32_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_32_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_32_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_32_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_32_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_32_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_32_drv	r/w	0	GPIO Driving Control
1	reg_gpio_32_smt	r/w	1	GPIO SMT Control
0	reg_gpio_32_ie	r/w	0	GPIO Input Enable

4.8.34 gpio_cfg33

Address: 0x20000948

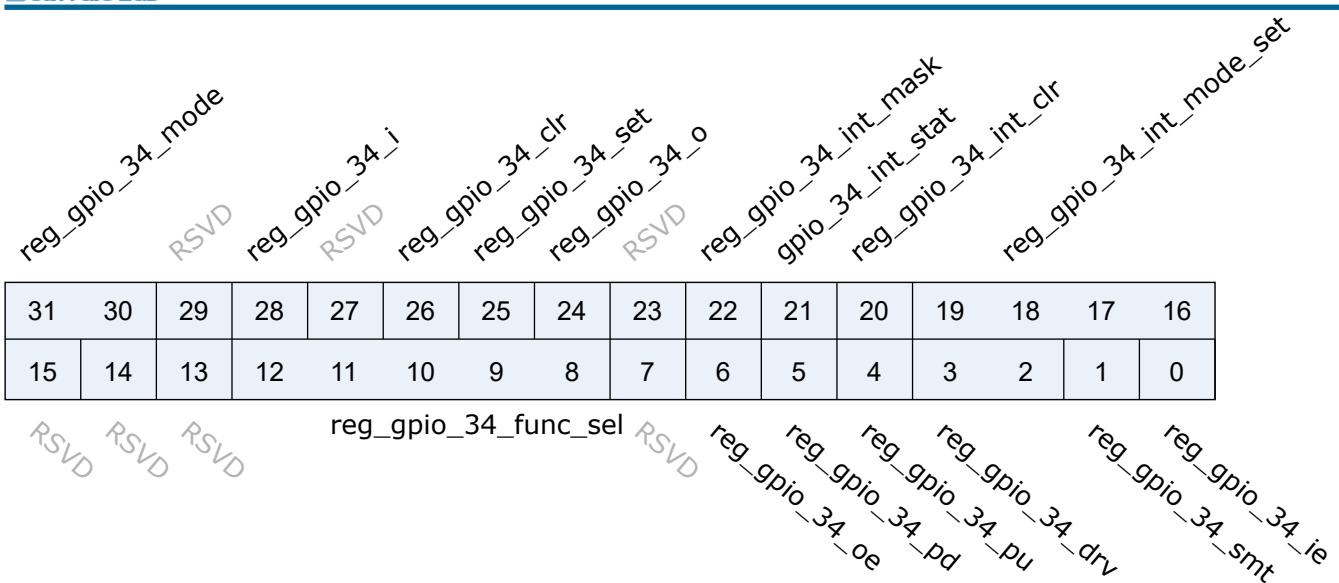


Bits	Name	Type	Reset	Description
31:30	reg_gpio_33_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_33_i	r	0	
27	RSVD			
26	reg_gpio_33_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_33_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_33_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_33_int_mask	r/w	1	mask interrupt (1)
21	gpio_33_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_33_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_33_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_33_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_33_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_33_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_33_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_33_drv	r/w	0	GPIO Driving Control
1	reg_gpio_33_smt	r/w	1	GPIO SMT Control
0	reg_gpio_33_ie	r/w	0	GPIO Input Enable

4.8.35 gpio_cfg34

Address: 0x2000094c



Bits	Name	Type	Reset	Description
31:30	reg_gpio_34_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Clear Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_34_i	r	0	
27	RSVD			
26	reg_gpio_34_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_34_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_34_o	r/w	0	When SWGPIO @ Output Value Mode 00 : GPIO Value changes according to this value 01 : GPIO Value Set by this register and clr by clr_reg
23	RSVD			
22	reg_gpio_34_int_mask	r/w	1	mask interrupt (1)
21	gpio_34_int_stat	r	0	interrupt status

Bits	Name	Type	Reset	Description
20	reg_gpio_34_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_34_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_34_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_34_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO)
5	reg_gpio_34_pd	r/w	0	GPIO Pull Down Control
4	reg_gpio_34_pu	r/w	0	GPIO Pull Up Control
3:2	reg_gpio_34_drv	r/w	0	GPIO Driving Control
1	reg_gpio_34_smt	r/w	1	GPIO SMT Control
0	reg_gpio_34_ie	r/w	0	GPIO Input Enable

4.8.36 gpio_cfg128

Address: 0x20000ac4

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
reg2_gpio_31_i	reg2_gpio_30_i	reg2_gpio_29_i	reg2_gpio_28_i	reg2_gpio_27_i	reg2_gpio_26_i	reg2_gpio_25_i	reg2_gpio_24_i	reg2_gpio_23_i	reg2_gpio_22_i	reg2_gpio_21_i	reg2_gpio_20_i	reg2_gpio_19_i	reg2_gpio_18_i	reg2_gpio_17_i	reg2_gpio_16_i

reg2_gpio_15_i	reg2_gpio_14_i	reg2_gpio_13_i	reg2_gpio_12_i	reg2_gpio_11_i	reg2_gpio_10_i	reg2_gpio_9_i	reg2_gpio_8_i	reg2_gpio_7_i	reg2_gpio_6_i	reg2_gpio_5_i	reg2_gpio_4_i	reg2_gpio_3_i	reg2_gpio_2_i	reg2_gpio_1_i	reg2_gpio_0_i
----------------	----------------	----------------	----------------	----------------	----------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------

Bits	Name	Type	Reset	Description
31	reg2_gpio_31_i	r	0	Register Controlled GPIO Input value
30	reg2_gpio_30_i	r	0	Register Controlled GPIO Input value
29	reg2_gpio_29_i	r	0	Register Controlled GPIO Input value
28	reg2_gpio_28_i	r	0	Register Controlled GPIO Input value
27	reg2_gpio_27_i	r	0	Register Controlled GPIO Input value
26	reg2_gpio_26_i	r	0	Register Controlled GPIO Input value
25	reg2_gpio_25_i	r	0	Register Controlled GPIO Input value
24	reg2_gpio_24_i	r	0	Register Controlled GPIO Input value
23	reg2_gpio_23_i	r	0	Register Controlled GPIO Input value
22	reg2_gpio_22_i	r	0	Register Controlled GPIO Input value
21	reg2_gpio_21_i	r	0	Register Controlled GPIO Input value
20	reg2_gpio_20_i	r	0	Register Controlled GPIO Input value
19	reg2_gpio_19_i	r	0	Register Controlled GPIO Input value
18	reg2_gpio_18_i	r	0	Register Controlled GPIO Input value
17	reg2_gpio_17_i	r	0	Register Controlled GPIO Input value
16	reg2_gpio_16_i	r	0	Register Controlled GPIO Input value
15	reg2_gpio_15_i	r	0	Register Controlled GPIO Input value
14	reg2_gpio_14_i	r	0	Register Controlled GPIO Input value
13	reg2_gpio_13_i	r	0	Register Controlled GPIO Input value
12	reg2_gpio_12_i	r	0	Register Controlled GPIO Input value
11	reg2_gpio_11_i	r	0	Register Controlled GPIO Input value
10	reg2_gpio_10_i	r	0	Register Controlled GPIO Input value
9	reg2_gpio_9_i	r	0	Register Controlled GPIO Input value
8	reg2_gpio_8_i	r	0	Register Controlled GPIO Input value
7	reg2_gpio_7_i	r	0	Register Controlled GPIO Input value
6	reg2_gpio_6_i	r	0	Register Controlled GPIO Input value
5	reg2_gpio_5_i	r	0	Register Controlled GPIO Input value
4	reg2_gpio_4_i	r	0	Register Controlled GPIO Input value
3	reg2_gpio_3_i	r	0	Register Controlled GPIO Input value
2	reg2_gpio_2_i	r	0	Register Controlled GPIO Input value
1	reg2_gpio_1_i	r	0	Register Controlled GPIO Input value

Bits	Name	Type	Reset	Description
0	reg2_gpio_0_i	r	0	Register Controlled GPIO Input value

4.8.37 gpio_cfg129

Address: 0x20000ac8

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

RSVD *RSVD*

reg2_gpio_34_i *reg2_gpio_33_i* *reg2_gpio_32_i*

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	reg2_gpio_34_i	r	0	Register Controlled GPIO Input value
1	reg2_gpio_33_i	r	0	Register Controlled GPIO Input value
0	reg2_gpio_32_i	r	0	Register Controlled GPIO Input value

4.8.38 gpio_cfg136

Address: 0x20000ae4

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

reg2_gpio_31_o *reg2_gpio_30_o* *reg2_gpio_29_o* *reg2_gpio_28_o* *reg2_gpio_27_o* *reg2_gpio_26_o* *reg2_gpio_25_o* *reg2_gpio_24_o* *reg2_gpio_23_o* *reg2_gpio_22_o* *reg2_gpio_21_o* *reg2_gpio_20_o* *reg2_gpio_19_o* *reg2_gpio_18_o* *reg2_gpio_17_o* *reg2_gpio_16_o*

reg2_gpio_15_o *reg2_gpio_14_o* *reg2_gpio_13_o* *reg2_gpio_12_o* *reg2_gpio_11_o* *reg2_gpio_10_o* *reg2_gpio_9_o* *reg2_gpio_8_o* *reg2_gpio_7_o* *reg2_gpio_6_o* *reg2_gpio_5_o* *reg2_gpio_4_o* *reg2_gpio_3_o* *reg2_gpio_2_o* *reg2_gpio_1_o* *reg2_gpio_0_o*

Bits	Name	Type	Reset	Description
31	reg2_gpio_31_o	r/w	0	Register Controlled GPIO Output Value
30	reg2_gpio_30_o	r/w	0	Register Controlled GPIO Output Value
29	reg2_gpio_29_o	r/w	0	Register Controlled GPIO Output Value
28	reg2_gpio_28_o	r/w	0	Register Controlled GPIO Output Value
27	reg2_gpio_27_o	r/w	0	Register Controlled GPIO Output Value
26	reg2_gpio_26_o	r/w	0	Register Controlled GPIO Output Value
25	reg2_gpio_25_o	r/w	0	Register Controlled GPIO Output Value
24	reg2_gpio_24_o	r/w	0	Register Controlled GPIO Output Value
23	reg2_gpio_23_o	r/w	0	Register Controlled GPIO Output Value
22	reg2_gpio_22_o	r/w	0	Register Controlled GPIO Output Value
21	reg2_gpio_21_o	r/w	0	Register Controlled GPIO Output Value
20	reg2_gpio_20_o	r/w	0	Register Controlled GPIO Output Value
19	reg2_gpio_19_o	r/w	0	Register Controlled GPIO Output Value
18	reg2_gpio_18_o	r/w	0	Register Controlled GPIO Output Value
17	reg2_gpio_17_o	r/w	0	Register Controlled GPIO Output Value
16	reg2_gpio_16_o	r/w	0	Register Controlled GPIO Output Value
15	reg2_gpio_15_o	r/w	0	Register Controlled GPIO Output Value
14	reg2_gpio_14_o	r/w	0	Register Controlled GPIO Output Value
13	reg2_gpio_13_o	r/w	0	Register Controlled GPIO Output Value
12	reg2_gpio_12_o	r/w	0	Register Controlled GPIO Output Value
11	reg2_gpio_11_o	r/w	0	Register Controlled GPIO Output Value
10	reg2_gpio_10_o	r/w	0	Register Controlled GPIO Output Value
9	reg2_gpio_9_o	r/w	0	Register Controlled GPIO Output Value
8	reg2_gpio_8_o	r/w	0	Register Controlled GPIO Output Value
7	reg2_gpio_7_o	r/w	0	Register Controlled GPIO Output Value
6	reg2_gpio_6_o	r/w	0	Register Controlled GPIO Output Value
5	reg2_gpio_5_o	r/w	0	Register Controlled GPIO Output Value
4	reg2_gpio_4_o	r/w	0	Register Controlled GPIO Output Value
3	reg2_gpio_3_o	r/w	0	Register Controlled GPIO Output Value
2	reg2_gpio_2_o	r/w	0	Register Controlled GPIO Output Value
1	reg2_gpio_1_o	r/w	0	Register Controlled GPIO Output Value

Bits	Name	Type	Reset	Description
0	reg2_gpio_0_o	r/w	0	Register Controlled GPIO Output Value

4.8.39 gpio_cfg137

Address: 0x20000ae8

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

RSVD *RSVD*

reg2_gpio_34_o *reg2_gpio_33_o* *reg2_gpio_32_o*

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	reg2_gpio_34_o	r/w	0	Register Controlled GPIO Output Value
1	reg2_gpio_33_o	r/w	0	Register Controlled GPIO Output Value
0	reg2_gpio_32_o	r/w	0	Register Controlled GPIO Output Value

4.8.40 gpio_cfg138

Address: 0x20000aec

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

reg2_gpio_31_set *reg2_gpio_30_set* *reg2_gpio_29_set* *reg2_gpio_28_set* *reg2_gpio_27_set* *reg2_gpio_26_set* *reg2_gpio_25_set* *reg2_gpio_24_set* *reg2_gpio_23_set* *reg2_gpio_22_set* *reg2_gpio_21_set* *reg2_gpio_20_set* *reg2_gpio_19_set* *reg2_gpio_18_set* *reg2_gpio_17_set* *reg2_gpio_16_set*

reg2_gpio_15_set *reg2_gpio_14_set* *reg2_gpio_13_set* *reg2_gpio_12_set* *reg2_gpio_11_set* *reg2_gpio_10_set* *reg2_gpio_9_set* *reg2_gpio_8_set* *reg2_gpio_7_set* *reg2_gpio_6_set* *reg2_gpio_5_set* *reg2_gpio_4_set* *reg2_gpio_3_set* *reg2_gpio_2_set* *reg2_gpio_1_set* *reg2_gpio_0_set*

Bits	Name	Type	Reset	Description
31	reg2_gpio_31_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
30	reg2_gpio_30_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
29	reg2_gpio_29_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
28	reg2_gpio_28_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
27	reg2_gpio_27_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
26	reg2_gpio_26_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
25	reg2_gpio_25_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg2_gpio_24_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
23	reg2_gpio_23_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
22	reg2_gpio_22_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
21	reg2_gpio_21_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
20	reg2_gpio_20_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
19	reg2_gpio_19_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
18	reg2_gpio_18_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
17	reg2_gpio_17_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
16	reg2_gpio_16_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
15	reg2_gpio_15_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
14	reg2_gpio_14_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
13	reg2_gpio_13_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
12	reg2_gpio_12_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
11	reg2_gpio_11_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
10	reg2_gpio_10_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
9	reg2_gpio_9_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
8	reg2_gpio_8_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
7	reg2_gpio_7_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
6	reg2_gpio_6_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
5	reg2_gpio_5_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
4	reg2_gpio_4_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
3	reg2_gpio_3_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
2	reg2_gpio_2_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
1	reg2_gpio_1_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
0	reg2_gpio_0_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect

4.8.41 gpio_cfg139

Address: 0x20000af0

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | | | |

reg2_gpio_34_set reg2_gpio_33_set reg2_gpio_32_set

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	reg2_gpio_34_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
1	reg2_gpio_33_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
0	reg2_gpio_32_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
-1:32	RSVD			
31	reg2_gpio_31_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
30	reg2_gpio_30_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
29	reg2_gpio_29_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
28	reg2_gpio_28_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
27	reg2_gpio_27_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
26	reg2_gpio_26_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg2_gpio_25_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
24	reg2_gpio_24_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
23	reg2_gpio_23_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
22	reg2_gpio_22_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
21	reg2_gpio_21_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
20	reg2_gpio_20_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
19	reg2_gpio_19_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
18	reg2_gpio_18_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
17	reg2_gpio_17_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
16	reg2_gpio_16_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
15	reg2_gpio_15_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
14	reg2_gpio_14_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
13	reg2_gpio_13_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
12	reg2_gpio_12_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
11	reg2_gpio_11_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
10	reg2_gpio_10_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
9	reg2_gpio_9_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
8	reg2_gpio_8_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
7	reg2_gpio_7_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
6	reg2_gpio_6_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
5	reg2_gpio_5_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
4	reg2_gpio_4_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
3	reg2_gpio_3_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
2	reg2_gpio_2_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
1	reg2_gpio_1_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
0	reg2_gpio_0_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

4.8.42 gpio_cfg141

Address: 0x20000af8

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	reg2_gpio_34_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
1	reg2_gpio_33_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
0	reg2_gpio_32_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

4.8.43 gpio_cfg142

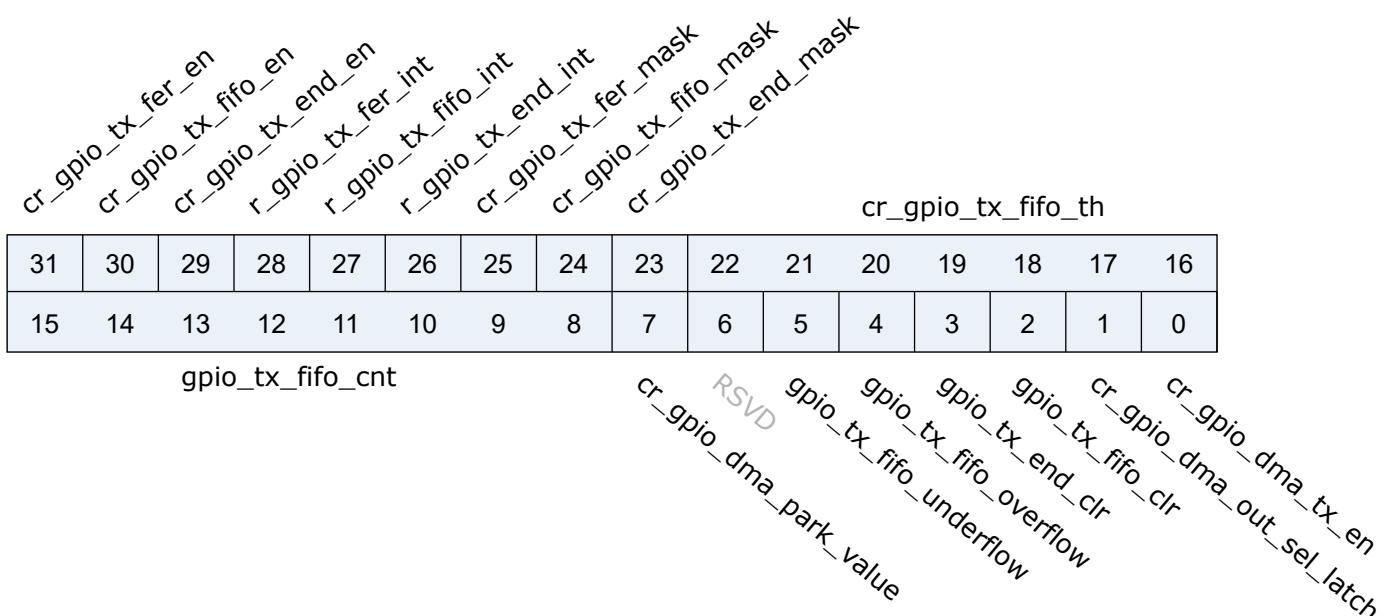
Address: 0x20000afc

cr_code1_high_time								cr_code0_high_time							
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
cr_code_total_time															
								RSVD	RSVD	RSVD	RSVD	cr_invert_code1_high	cr_invert_code0_high	cr_gpio_tx_en	

Bits	Name	Type	Reset	Description
31:24	cr_code1_high_time	r/w	8'd200	Used to generate Code 1 Duty Cycle Waveform (in units of xclk (XTAL or RC32M) clock cycle) waveform keep high during cr_code1_high_time waveform keep low during cr_code1_low_time (cr_code_total_time - cr_code1_high_time)
23:16	cr_code0_high_time	r/w	8'd200	Used to generate Code 0 Duty Cycle Waveform (in units of xclk (XTAL or RC32M) clock cycle) waveform keep high during cr_code0_high_time waveform keep low during cr_code0_low_time (cr_code_total_time - cr_code0_high_time)
15:7	cr_code_total_time	r/w	9'd400	Used to define Code0/Code1 total waveform time
6:3	RSVD			
2	cr_invert_code1_high	r/w	1'b0	1: cr_code1_high_time -> cr_code1_low_time
1	cr_invert_code0_high	r/w	1'b0	1: cr_code0_high_time -> cr_code0_low_time
0	cr_gpio_tx_en	r/w	1'b0	Enable GPIO DMA OUT/GPIO DMA Latch

4.8.44 gpio_cfg143

Address: 0x20000b00



Bits	Name	Type	Reset	Description
31	cr_gpio_tx_fer_en	r/w	1'b1	Interrupt enable of gpio_tx_fer_int (GPIO DMA FIFO Underflow or Overflow)
30	cr_gpio_tx_fifo_en	r/w	1'b1	Interrupt enable of gpio_tx_fifo_int
29	cr_gpio_tx_end_en	r/w	1'b1	Interrupt enable of gpio_tx_end_int
28	r_gpio_tx_fer_int	r	1'b0	GPIO TX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
27	r_gpio_tx_fifo_int	r	1'b0	GPIO TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
26	r_gpio_tx_end_int	r	1'b0	GPIO TX END Interrupt (GPIO DMA FIFO Empty)
25	cr_gpio_tx_fer_mask	r/w	1'b1	Interrupt mask of gpio_tx_fer_int
24	cr_gpio_tx_fifo_mask	r/w	1'b1	Interrupt mask of gpio_tx_fifo_int
23	cr_gpio_tx_end_mask	r/w	1'b1	Interrupt mask of gpio_tx_end_int
22:16	cr_gpio_tx_fifo_th	r/w	7'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:8	gpio_tx_fifo_cnt	r	8'd128	TX FIFO available count
7	cr_gpio_dma_park_value	r/w	1'b0	1: park at high level when TX FIFO empty 0: park at low level when TX FIFO empty
6	RSVD			
5	gpio_tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	gpio_tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	gpio_tx_end_clr	w1c	1'b0	Interrupt clear of gpio_tx_end_int
2	gpio_tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	cr_gpio_dma_out_sel_latch	r/w	1'b0	1 : select latch format (8bit set/8bit clr) , 0 : select 16bit output value
0	cr_gpio_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

4.8.45 gpio_cfg144

Address: 0x20000b04

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | |

gpio_tx_data_to_fifo

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	gpio_tx_data_to_fifo	w	x	

5.1 Overview

The chip integrates one 12-bit SAR ADC, which supports 12 external analog input signals and several internal analog signals. ADC can work in single conversion mode and multi-channel scanning mode, and the conversion result is 12/14/16-bit LeftJustified mode. ADC has a FIFO with a depth of 32, which supports multiple interrupts and DMA. In addition to measuring ordinary analog signals, ADC can also measure power supply voltage, and detect temperature by measuring the internal/external diode voltage.

5.2 Features

- High performance
 - Optional 12-bit/14-bit/16-bit conversion result for output
 - Shortest ADC conversion time of 0.5 us (12-bit conversion result)
 - Optional reference voltage of 2.0V/3.3V
 - Transfer of conversion result to memory through DMA
 - Supports single channel conversion and multi-channel scanning modes
 - Single-ended and differential input modes
 - Jitter compensation
 - User-defined offset value of conversion result
 - Up to 1M for scanning mode clock and 2M for non-scanning mode clock
- Number of analog channels
 - Twelve external analog channels
 - Two internal DAC channels

- One VBAT/2 channel
- One TSEN channel

5.3 Functional Description

The block diagram of ADC module is shown as follows.

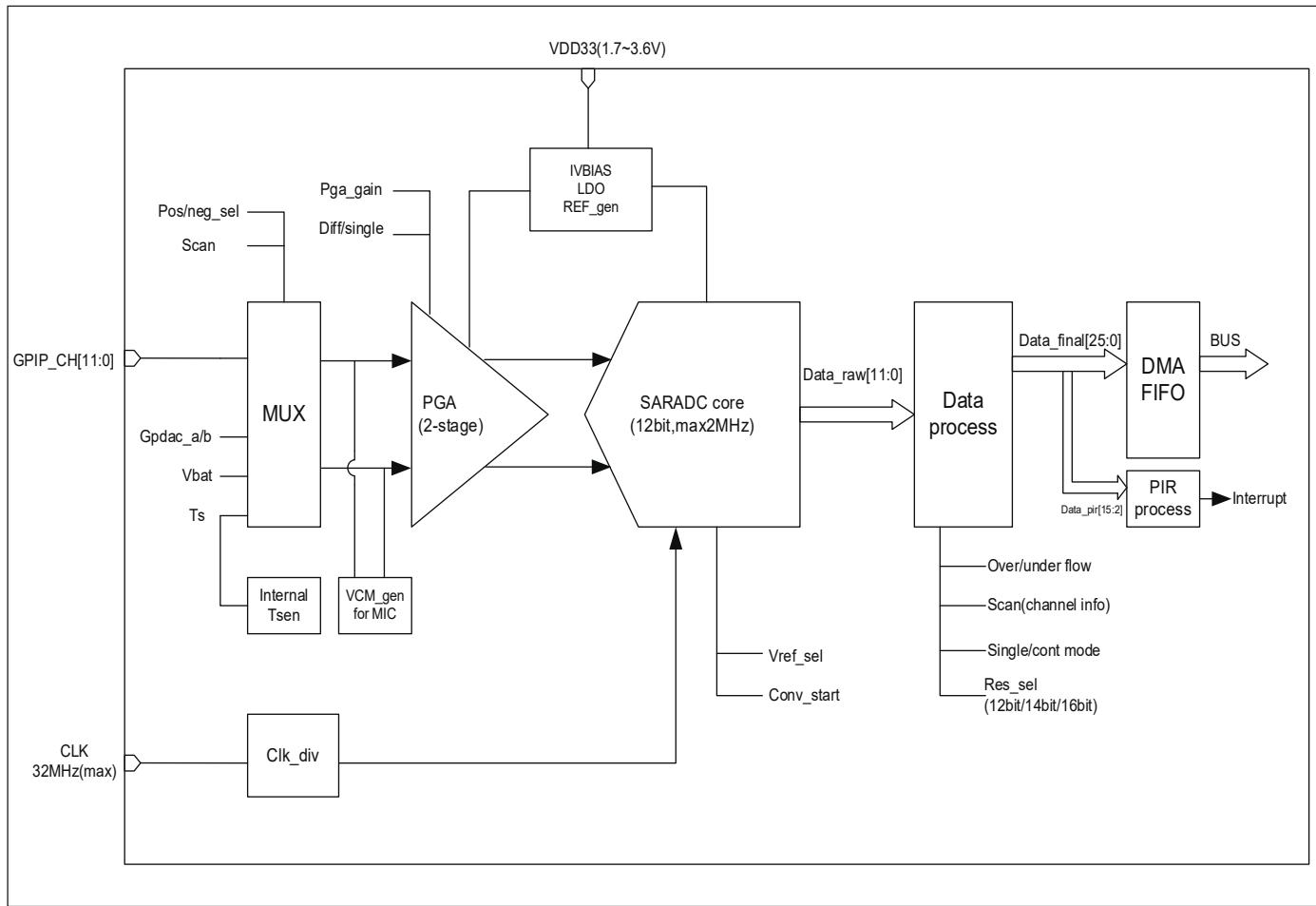


Fig. 5.1: Block Diagram of ADC

The ADC module consists of front-end input channel selector, programmable amplifier, ADC sampling module, data processing module, and FIFO.

The input channel selector is used to select the channel to be sampled, including internal and external analog signals.

The programmable amplifier is used to further process the input signal, and it is set based on the characteristics of the input signal (DC/AC) to get a more accurate conversion value.

The ADC sampling module is the most important functional module, which gets the result of conversion (12 bit) from analog signal to digital signal by successive comparison.

The data processing module further processes the conversion result, including adding channel information.

The final data will be pushed to the back-end FIFO.

5.3.1 ADC Pins and Internal Signals

Table 5.1: ADC internal signal

Internal Signal	Type	Description
VBAT/2	Input	Voltage signal divided from the power supply pin
TSEN	Input	Output voltage of internal temperature sensor
VREF	Input	Reference voltage of internal analog module
DACOUTA	Input	DAC module output
DACOUTB	Input	DAC module output

Table 5.2: ADC external pin

External Pin	Signal Type	Signal Description
VDDA	Input	Positive supply voltage of analog module
VSSA	Input	Analog module ground
ADC_CHX	Input	Analog input pins, 12 channels

5.3.2 ADC Channels

The selectable channels of ADC sampling include the input signals of external analog pins and the selectable signals inside the chip:

- ADC CH0
- ADC CH1
- ADC CH2
- ADC CH3
- ADC CH4
- ADC CH5
- ADC CH6
- ADC CH7
- ADC CH8

- ADC CH9
- ADC CH10
- ADC CH11
- DAC OUTA
- DAC OUTB
- VBAT/2
- TSEN
- VREF
- GND

It is worth noting that if VBAT/2 or TSEN is selected as the input signal to be sampled, gpadc_vbat_en or gpadc_ts_en shall be set. The ADC module supports single-ended input or differential input. For the former, GND shall be selected as the negative input channel.

5.3.3 ADC Clocks

The following figure illustrates the working clock source of the ADC module.

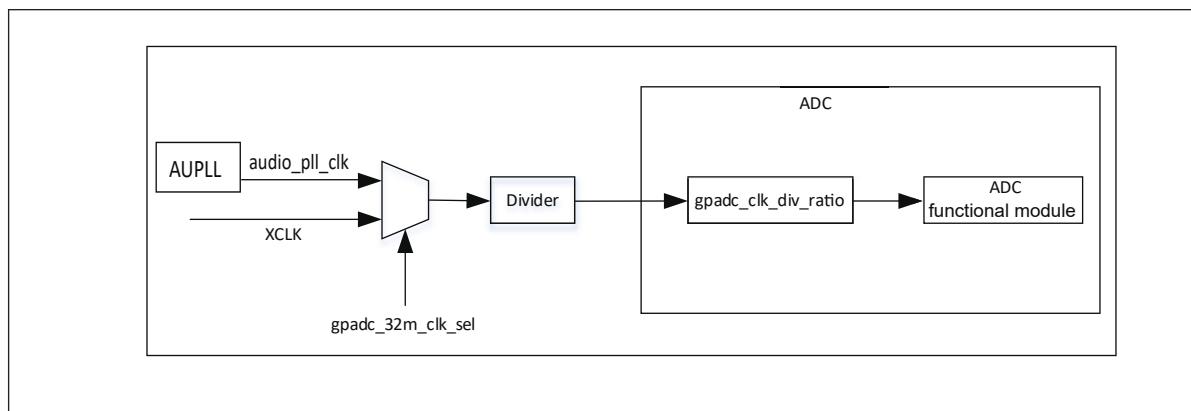


Fig. 5.2: ADC Clocks

The clock source of ADC can be the audio_pll_clk from AUPLL, XTAL or internal RC32M. The GLB module controls the setting of clock source selection and provides clock frequency division. For example: the clock source of ADC is XTAL and the clock frequency division is 0. The clock reaching the ADC module is 40M.

Inside the ADC module, a clock divider is provided. If you choose to divide by 20, the clock inside the ADC module is 2M. Users can adjust the ADC clock source and various frequency division coefficients by themselves according to the actual sampling requirements.

The gpadc_32m_clk_div frequency division register is 6 bits wide (max div=64). The frequency division formula is $f_{out} = f_{source} / (gpadc_32m_clk_div + 1)$. The gpadc_clk_div_ratio frequency division register of 3 bits width is located

inside the ADC module. Its frequency division values are defined as follows:

- 3' b000: div=1
- 3' b001: div=4
- 3' b010: div=8
- 3' b011: div=12
- 3' b100: div=16
- 3' b101: div=20
- 3' b110: div=24
- 3' b111: div=32

5.3.4 ADC Conversion Mode

The ADC supports single-channel conversion and scan conversion. In single-channel conversion mode, the user needs to select the positive input channel through gpadc_pos_sel, select the negative input channel through gpadc_neg_sel, and set the gpadc_cont_conv_en control bit to 0, indicating single-channel conversion, then set the gpadc_conv_start control bit to start the conversion.

In the scanning conversion mode, the gpadc_cont_conv_en control bit is set to 1. ADC performs conversion one by one according to the number of conversion channels set by the gpadc_scan_length control bit and the channel sequence set by gpadc_reg_scn_posX (X = 1, 2) and gpadc_reg_scn_negX (X = 1, 2) register sets. The conversion result is automatically pushed into the ADC FIFO. The channels set by the gpadc_reg_scn_posX (X = 1, 2) and gpadc_reg_scn_negX (X = 1, 2) register sets can be the same, which means that the user can sample a channel multiple times for conversion.

The conversion result of ADC is usually pushed into FIFO. Users need to set the FIFO data receiving threshold interrupt according to the actual number of conversion channels, and this threshold interrupt is used as the ADC Conversion Complete Interrupt.

5.3.5 ADC Result

The register gpadc_raw_data stores the original result of ADC. In the single-ended mode, the valid bit of data is 12 bits, without sign bit. In the differential mode, the most significant bit (MSB) is the sign bit, and the remaining 11 bits represent the conversion result.

The register gpadc_data_out stores the ADC result, which contains the ADC result, sign bit, and channel information. The data format is as follows:

Table 5.3: Meaning of ADC Conversion Result

BitS	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			
meaning	Positive channel number					Negative channel number					Conversion result																		

In the above table, bit21-bit25 indicates the positive channel number, and bit16-bit20 indicates the negative channel number, while bit0-bit15 indicates the converted value.

The gpadc_res_sel control bit can set the bits of the conversion result to 12 bits, 14 bits, and 16 bits, of which 14 bits and 16 bits are the result of multiple sampling. The optional setting values are as follows:

- 3' b000 12bit 2MS/s, OSR=1
- 3' b001 14bit 125kS/s, OSR=16
- 3' b010 14bit 31.25kS/s, OSR=64
- 3' b011 16bit 15.625KS/s, OSR=128
- 3' b100 16bit 7.8125KS/s, OSR=256

In the left-justified ADC conversion result, when 12 bits are selected, bit15-bit4 of the conversion result is valid; when 14 bits are selected, bit15-bit2 is valid; and when 16 bits are selected, bit15-bit0 is valid. Similarly, in the differential mode, the MSB is the sign bit. Namely, when 14 bits are selected, bit15 is the sign bit and bit14 is the MSB, while bit14-bit2 is the conversion result. In the single-ended mode, there is no sign bit. Namely, when 12 bits are selected, bit15-bit4 is the conversion result and bit15 is the MSB.

In practice, ADC results are generally pushed into FIFO, which is especially important in the multi-channel scanning mode. Therefore, users usually obtain conversion results from ADC FIFO. The data format of ADC FIFO is the same as that in the register gpadc_data_out.

5.3.6 ADC Interrupt

The ADC module will generate interrupts upon positive or negative sampling over-range, and such interrupts can be masked by gpadc_pos_satur_mask and gpadc_neg_satur_mask. When interrupts are generated, the interrupt status can be queried through the registers gpadc_pos_satur and gpadc_neg_satur. Interrupts can be cleared through gpadc_pos_satur_clr and gpadc_neg_satur_clr. This function can be used to check whether the input voltage is abnormal.

5.3.7 ADC FIFO

The ADC module has a FIFO with a depth of 32 and its data width is 26 bits. When the ADC completes conversion, the result will be automatically pushed into the FIFO. ADC FIFO has the following statuses and interrupt management functions:

- FIFO: full
- FIFO: non-empty
- FIFO Overrun interrupt
- FIFO Underrun interrupt

When an interrupt is generated, the interrupt flag can be cleared by the corresponding clear bit.

ADC FIFO enables users to obtain data through query, interrupt, and DMA modes.

Query mode

CPU polls the gpadc_rdy bit: When the control bit is set, it indicates that there are valid data in FIFO. CPU can get the amount of FIFO data through gpadc_fifo_data_count and read out these data from FIFO.

Interrupt mode

If gpadc_rdy_mask is set to 0 in CPU, ADC will generate an interrupt when data is pushed into FIFO. In the interrupt function, the user can get the amount of FIFO data through gpadc_fifo_data_count and read out these data from FIFO, and then set gpadc_rdy_clr to clear the interrupt.

DMA mode

If the user sets the gpadc_dma_en control bit, DMA can transfer the conversion data to the memory. In the DMA mode, after the data volume threshold is set for ADC FIFO to send DMA requests through gpadc_fifo_thl, when receiving a request, DMA will automatically transfer the specified number of results from FIFO to the corresponding memory according to the user defined parameters.

5.3.8 ADC Setup Process

Set ADC clock

According to the required ADC conversion speed, determine the working clock of ADC, set the ADC clock source and frequency division of the GLB module, and determine the final working clock frequency of the ADC module based on the gpadc_clk_div_ratio.

Set GPIO according to the channel used

According to the analog pin used, determine the channel number used, and initialize the corresponding GPIO to analog function. But note that when setting GPIO as analog input, set it as floating input instead of pull-up or pull-down.

Set the channel to convert

According to the used analog channel and conversion mode, set the corresponding channel register. For single channel conversion, set the conversion channel information in the registers gpadc_pos_sel and gpadc_neg_sel. In the multi-channel scanning mode, set gpadc_scan_length, gpadc_reg_scn_posX, and gpadc_reg_scn_negX according to the number of channels to be scanned and the scanning sequence.

Set the data read method

Based on the data reading mode introduced by ADC FIFO, select the mode used and set the register. If DMA is used, configure one channel of DMA, and assist ADC FIFO in transferring data.

Start conversion

Finally, set gpadc_res_sel to select the precision of data conversion result, and then set gpadc_global_en=1 and gpadc_conv_start=1 to start ADC conversion. When conversion is completed, to convert again, it is necessary to set gpadc_conv_start to 0 first and then to 1 to trigger the conversion again.

5.3.9 VBAT Measurement

The VBAT/2 here measures the voltage of the chip VDD33, not the external lithium battery voltage. If you need to measure the voltage of power supply sources like lithium battery, you can divide the voltage and input it into the GPIO analog channel of ADC. Measuring the voltage of VDD33 can reduce the use of GPIO.

The VBAT/2 voltage measured by the ADC module is divided, and the actual voltage input to the ADC module is half of that of VDD33, namely $VBAT/2=VDD33/2$. As the voltage is divided, to ensure a high accuracy, it is suggested to select 2.0 V reference voltage for ADC, and enable the single-ended mode. The positive input voltage is set to VBAT/2 and the negative input voltage is set to GND. Gpadc_vbat_en is set to 1. After conversion starts, the corresponding conversion result can be multiplied by 2 to get the voltage of VDD33.

5.3.10 TSEN Measurement

ADC can measure the voltage of internal or external diode. As the voltage difference of diode is related to temperature, the ambient temperature can be calculated from the measured voltage of diode. So it is called temperature sensor (TSEN).

The measurement principle of TSEN is the curve fitted by the voltage difference (ΔV) with the change of temperature, where ΔV is produced by measuring two different currents on a diode. No matter an external or internal diode is measured, the final output value is related to temperature and can be expressed as $\Delta(ADC_{out})=7.753T+X$. As long as we know the voltage, we know the temperature T. X indicates an offset value, which can be used as a standard value. Before actual use, X shall be determined first. The chip manufacturer will measure $\Delta(ADC_{out})$ at a standard temperature, such as 25°C room temperature, before the chip leaves the factory, to obtain X. In actual use, the user can get the temperature T according to the formula $T=[\Delta(ADC_{out})X]/7.753$.

When TSEN is used, it is recommended to set ADC to the 16bits mode, reduce error through multiple sampling, select 1.8 V reference voltage to improve accuracy, and set gpadc_ts_en to 1 to enable the TSEN function. If internal diode is selected, gpadc_tsext_sel=0. If external diode is selected, gpadc_tsext_sel=1. The positive input channel is selected according to actual needs, namely TSEN channel for the internal diode or the analog GPIO channel for the

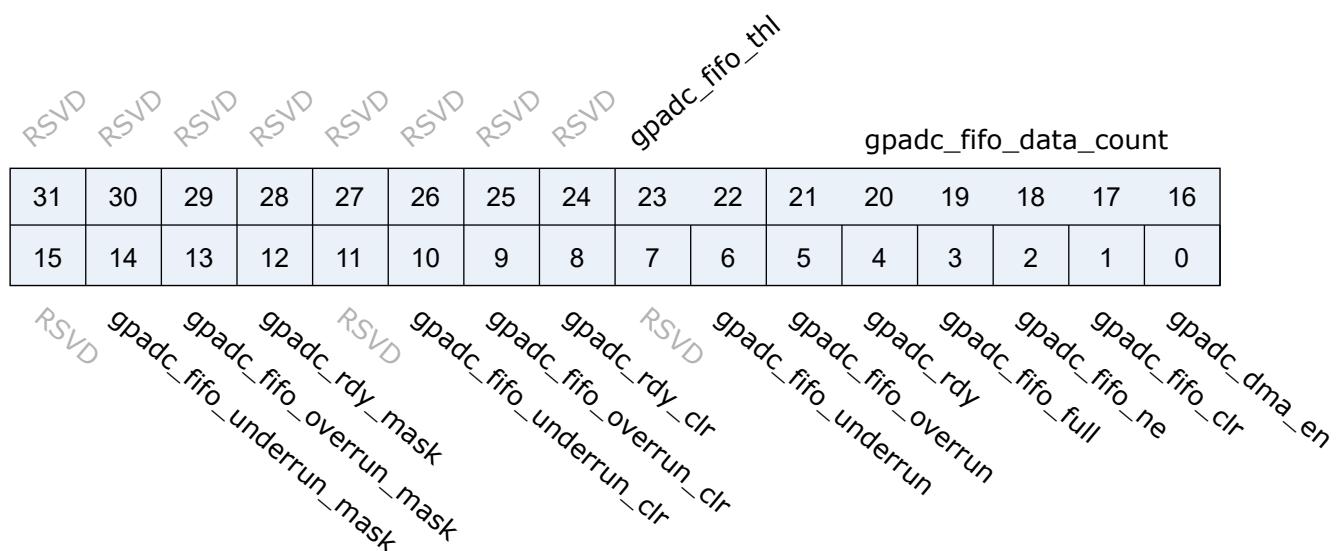
external diode. The negative input channel is set to GND. After finishing the above settings, set gpadc_tsvbe_low=0 to start measurement, and get the measurement result V0. Then set gpadc_tsvbe_low=1 to start measurement, and get the measurement result V1, $\Delta(\text{ADC_out})=V1-V0$. The temperature T is calculated based on the formula $T=[\Delta(\text{ADC_out})X]/7.753$.

5.4 Register description

Name	Description
gpadc_config	
gpadc_dma_rdata	

5.4.1 gpadc_config

Address: 0x20002000



Bits	Name	Type	Reset	Description
31:24	RSVD			
23:22	gpadc_fifo_thl	r/w	2'd0	fifo threshold 2'b00: 1 data 2'b01: 4 data 2'b10: 8 data 2'b11: 16 data
21:16	gpadc_fifo_data_count	r	6'd0	fifo data number
15	RSVD			

Bits	Name	Type	Reset	Description
14	gpadc_fifo_underrun_mask	r/w	1'b0	write 1 mask
13	gpadc_fifo_overrun_mask	r/w	1'b0	write 1 mask
12	gpadc_rdy_mask	r/w	1'b0	write 1 mask
11	RSVD			
10	gpadc_fifo_underrun_clr	r/w	1'b0	Write 1 to clear flag
9	gpadc_fifo_overrun_clr	r/w	1'b0	Write 1 to clear flag
8	gpadc_rdy_clr	r/w	1'b0	Write 1 to clear flag
7	RSVD			
6	gpadc_fifo_underrun	r	1'b0	FIFO underrun interrupt flag
5	gpadc_fifo_overrun	r	1'b0	FIFO overrun interrupt flag
4	gpadc_rdy	r	1'b0	Conversion data ready interrupt flag
3	gpadc_fifo_full	r	1'b0	FIFO full flag
2	gpadc_fifo_ne	r	1'b0	FIFO not empty flag
1	gpadc_fifo_clr	w1c	1'b0	FIFO clear signal
0	gpadc_dma_en	r/w	1'b0	GPADC DMA enable

5.4.2 gpadc_dma_rdata

Address: 0x20002004

gpadc_dma_rdata															
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
gpadc_dma_rdata															

Bits	Name	Type	Reset	Description
31:26	RSVD			
25:0	gpadc_dma_rdata	r	26'd0	GPADC final conversion result stored in the FIFO

6.1 Overview

The DAC module is a 12-bit voltage output digital-to-analog converter that works with a DMA controller. The DAC has two output channels, each with a converter, and each channel can be converted independently. Can be used for audio playback, transmitter voltage modulation and other applications.

6.2 Features

- DAC modulation accuracy is 12-bits
- DAC input clock can be selected as 32MHz or xclk
- Each channel has DMA function and supports 10 data transfer formats
- DAC dual channels convert individually or simultaneously
- Support two-channel playback DMA transfer mode
- The output pin of DAC is fixed as ChannelA is GPIO3, ChannelB is GPIO2
- Supports internal and external input reference voltages

6.3 Functional Description

The block diagram of DAC is shown as follows.

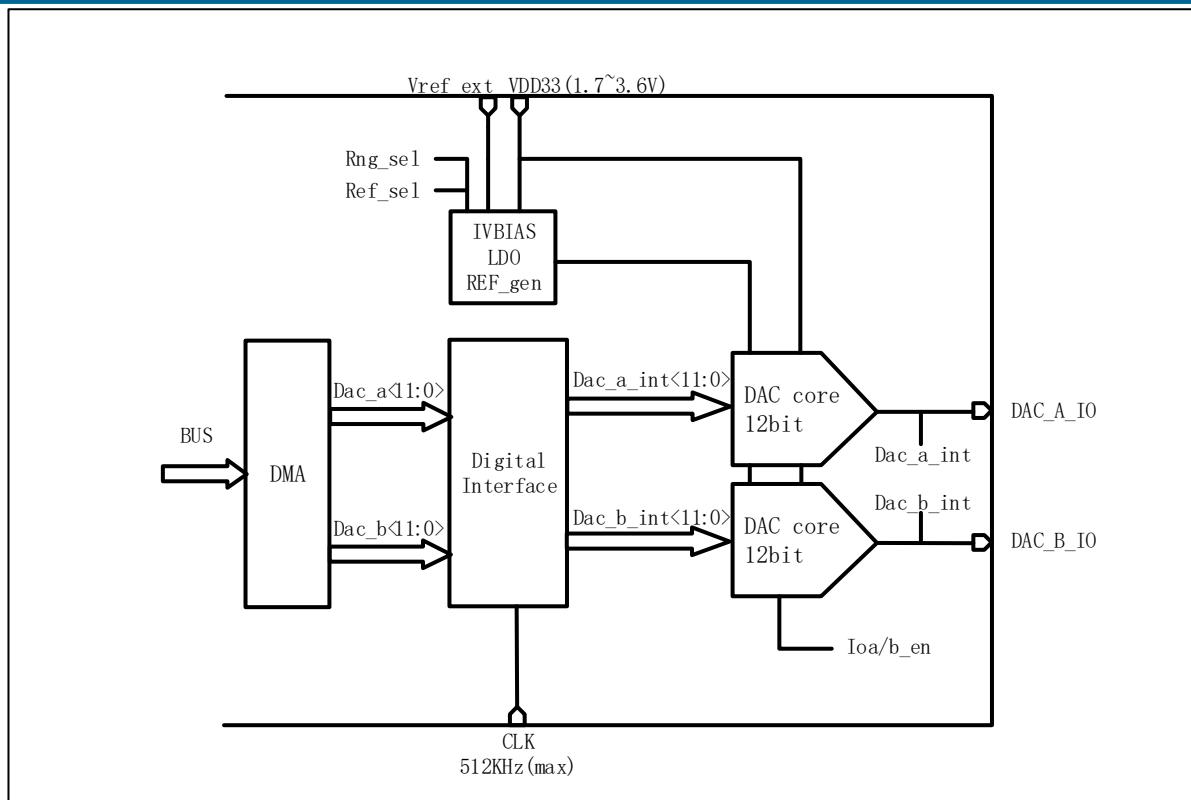


Fig. 6.1: Block Diagram of DAC

The DAC module includes two DAC modulation circuits and power supply circuits related to modulating analog signals. The user can select the reference voltage of the DAC through Ref_Sel to be external/internal, and Rng_Sel to select the output voltage range. The modulation data of the DAC can be directly written to the DAC modulation register (gpdac_a_data, gpdac_b_data in the register dac_cfg3) by the CPU, or transferred to the gpdac_dma_wdata register by the DMA.

6.3.1 DAC channel enable

Taking enabling channel A as an example, the configuration process is as follows:

1. Set the gpdac_en bit in register gpdac_config to 1 to enable the DAC
2. Enable DAC A channel conversion by setting the gpdac_a_en bit in register dac_cfg1 to 1
3. Set the gpdac_ioa_en bit in register dac_cfg1 to 1 to enable the conversion result of channel A to the GPIO port

6.3.2 DAC data format

When using DMA to convert data, there are a total of 10 data transmission formats, which can be configured by setting the corresponding gpdac_dma_format bit value in the register gpdac_dma_config. The data transmission format stored in the gpdac_dma_wdata register (with a width of 32-bit), the corresponding relationship is as follows:

Table 6.1: Data transfer format

Value of gpdac_dma_format	gpdac_dma_wdata(Corresponding to the data transmission format of A and B channels)
0	[11:0]: {A0}, {A1}, {A2} ...
1	[27:16][11:0]: {B0,A0}, {B1,A1}, {B2,A2} ...
2	[27:16][11:0]: {A1,A0}, {A3,A2}, {A5,A4} ...
4	[15:4]: {A0}, {A1}, {A2} ...
5	[31:20][15:4]: {B0,A0}, {B1,A1}, {B2,A2} ...
6	[31:20][15:4]: {A1,A0}, {A3,A2}, {A5,A4} ...
8	[31:24][23:16][15:8][7:0]: {A3,A2,A1,A0}, {A7,A6,A5,A4} ...
9	[31:24][23:16][15:8][7:0]: {B1,B0,A1,A0}, {B3,B2,A3,A2} ...
10	[31:24][23:16][15:8][7:0]: {B1,A1,B0,A0}, {B3,A3,B2,A2} ...
11	[15:8][7:0]: {B0,A0}, {B1,A1}, {B2,A2} ...

6.3.3 DAC output voltage

The user can choose to use the external reference voltage or the internal reference voltage by setting the corresponding gpdac_ref_sel bit value in the dac_cfg0 register.

If the internal reference voltage is selected, the configuration and output voltage are shown in the table below. If external reference voltage is selected, connect the external voltage to fixed GPIO28.

Table 6.2: Internal reference voltage output voltage

gpdac_a_rng	gpdac_ref_sel	Output range (V)
00	0	0.2-1
01/10	0	0.225-1.425
11	0	0.2-1.8

6.3.4 DAC conversion

6.3.4.1 CPU mode

Use the CPU method to perform data conversion, taking the simultaneous conversion of channel A and channel B as an example, the configuration process is as follows:

1. Set the DAC clock: write the value of the corresponding `dig_clk_src_sel` bit in the register `dig_clk_cfg0` to 1, that is, select `xclk` as the DAC clock source
2. Set the clock frequency division factor: the user sets the value of the `dig_512k_div` bit corresponding to the register `dig_clk_cfg0` and the `gpdac_mode` bit corresponding to the register `gpdac_config` according to the requirements
3. Initialize the GPIO pins of the A and B channels
4. Initialize and enable DAC channel A and channel B
5. Write the data to be converted into the corresponding `gpdac_a_data` and `gpdac_b_data` bits in register `dac_cfg3` to complete the data conversion

6.3.4.2 DMA method

Each DAC channel has DMA capability. Taking channel A using DMA for data conversion as an example, the configuration process is as follows:

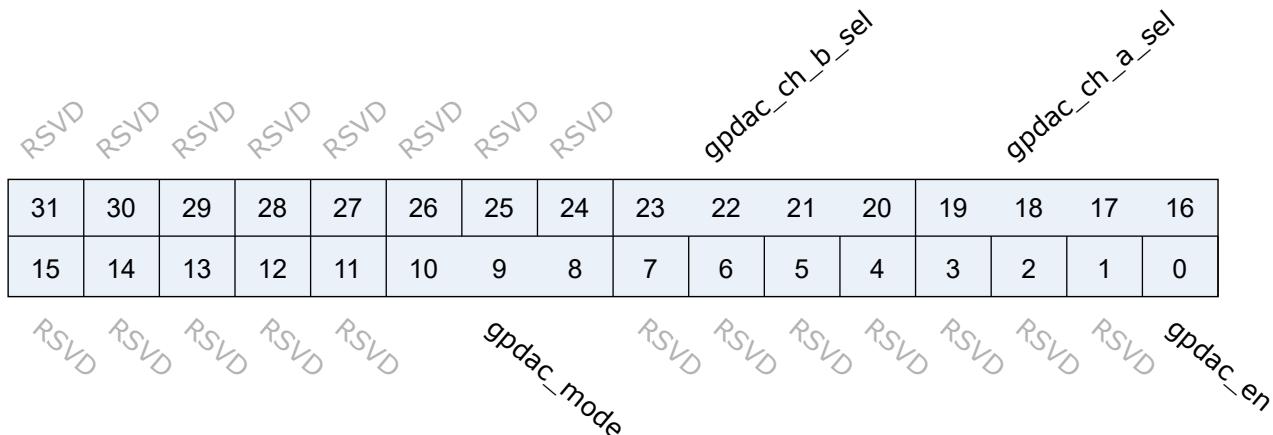
1. Set the DAC clock: write the value of the corresponding `dig_clk_src_sel` bit in the register `dig_clk_cfg0` to 1, that is, select `xclk` as the DAC clock source
2. Set the clock frequency division factor: the user sets the value of the `dig_512k_div` bit corresponding to the register `dig_clk_cfg0` and the `gpdac_mode` bit corresponding to the register `gpdac_config` according to the requirements
3. Initialize the GPIO pins of the A channel
4. Initialize and enable the DAC A channel
5. Initialize and enable the DMA channel: set the DMA transfer data width, source address, destination address and data transfer length, etc.
6. Enable DAC DMA mode: write 1 to the corresponding bit value of `gpdac_dma_tx_en` in register `gpdac_dma_config`
7. Write the data to be converted into the register `gpdac_dma_wdata`, and act on the A or B channel according to different data formats to complete the data conversion

6.4 Register description

Name	Description
gpdac_config	
gpdac_dma_config	
gpdac_dma_wdata	

6.4.1 gpdac_config

Address: 0x20002040

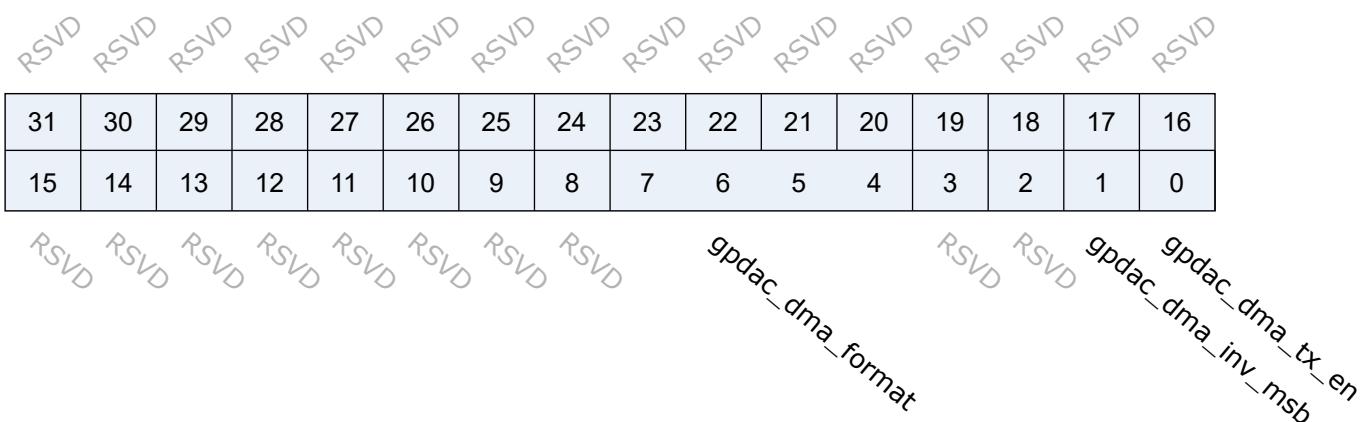


Bits	Name	Type	Reset	Description
31:24	RSVD			
23:20	gpdac_ch_b_sel	r/w	0	Channel B Source Select 0: Reg 1: DMA 2: DMA + Filter 3: Sin Gen 4: A (The same as channel A) 5: A (Inverse of channel A)
19:16	gpdac_ch_a_sel	r/w	0	Channel A Source Select 0: Reg 1: DMA 2: DMA + Filter 3: Sin Gen
15:11	RSVD			

Bits	Name	Type	Reset	Description
10:8	gpdac_mode	r/w	0	0:32k, 1:16k, 3:8k, 4:512k(for DMA only)
7:1	RSVD			
0	gpdac_en	r/w	0	GPDAC enable

6.4.2 gpdac_dma_config

Address: 0x20002044



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:4	gpdac_dma_format	r/w	0	DMA TX format (Data 12-bit) 0: ([11:0]) A0, A1, A2... 1: ([27:16][11:0]) B0,A0, B1,A1, B2,A2... 2: ([27:16][11:0]) A1,A0, A3,A2, A5,A4... 4: ([15:4]) A0, A1, A2... 5: ([31:20][15:4]) B0,A0, B1,A1, B2,A2... 6: ([31:20][15:4]) A1,A0, A3,A2, A5,A4... 8: ([31:24][23:16][15:8][7:0]) A3,A2,A1,A0, A7,A6,A5,A4... 9: ([31:24][23:16][15:8][7:0]) B1,B0,A1,A0, B3,B2,A3,A2... 10: ([31:24][23:16][15:8][7:0]) B1,A1,B0,A0, B3,A3,B2,A2... ... 11: ([15:8][7:0]) B0,A0, B1,A1, B2,A2...
3:2	RSVD			
1	gpdac_dma_inv_msb	r/w	0	GPDAC DMA Data Inverse MSB
0	gpdac_dma_tx_en	r/w	0	GPDAC DMA TX enable

6.4.3 gpdac_dma_wdata

Address: 0x20002048

gpdac_dma_wdata

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

gpdac_dma_wdata

Bits	Name	Type	Reset	Description
31:0	gpdac_dma_wdata	w	x	GPDAC DMA TX data

7.1 Overview

Direct Memory Access (DMA), a kind of memory access technique, can directly read and write system memory independently without relying on processor. Under the same processor load, DMA is a fast way to transfer data.

It is provided with 1 independent DMA controllers, of which DMA0 and DMA2 have 8 independent dedicated channels, and DMA1 has 4 such channels, managing data transmission between peripherals and memory to enhance bus efficiency. DMA supports the transfer from memory to memory, memory to peripherals, and peripheral to memory, and provides the LLI linked list function. The software configures the data transfer size, data source address, and destination address.

7.2 Features

- 1 DMA controller with 4 independent dedicated channels
- Independent control source and destination access width (single byte, double bytes, and four bytes)
- Each channel acts as a read-write cache independently
- Each channel can be triggered by independent peripheral hardware or software
- Eight kinds of process control
 - DMA process control, source memory, destination memory
 - DMA process control, source memory, destination peripherals
 - DMA process control, source peripherals, destination memory
 - DMA process control, source peripherals, destination peripherals
 - Destination peripheral process control, source peripherals, destination peripherals
 - Destination peripheral process control, source memory, destination peripherals

- Source peripheral process control, source peripherals, destination memory
- Source peripheral process control, source peripherals, destination peripherals
- Supports the LLI linked list function to improve DMA efficiency

7.3 Functional Description

7.3.1 Operating Principle

When a device tries to directly transfer data to another device through the bus, it will first send a DMA request signal to CPU. The peripheral submits a request through DMA to take over control of the bus from CPU. After receiving this signal, CPU will respond to the DMA signal according to the priority of signal and the order of the DMA request after the current bus cycle is over. When CPU responds to a DMA request for a device interface, it will give up control of the bus.

Under the control of DMA controllers, peripherals and memory exchange data directly without CPU intervention. After data is transferred, the device will send a DMA end signal to CPU and return the control of the bus.

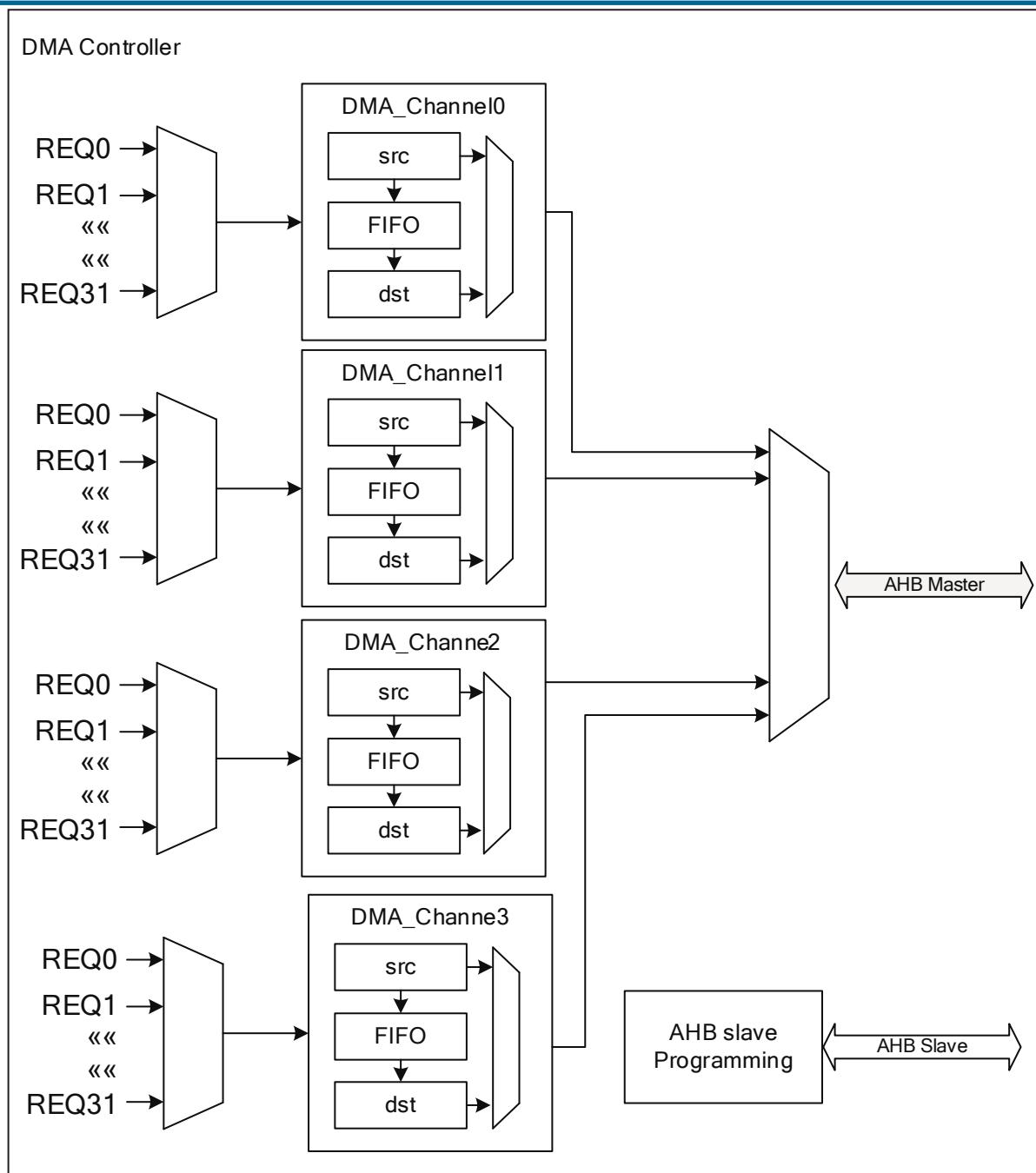


Fig. 7.1: Block Diagram of DMA

DMA has one AHB Master interfaces and one AHB Slave interface. Based on the current configuration requirements, the AHB Master interface actively accesses the memory or peripherals through system bus, and serves as a data transfer port. The AHB Slave interface for configuring DMA only supports 32bit access.

7.3.2 DMA Channel Configuration

DMA support 4 channels. All channels can run simultaneously without interference. Here is the configuration process of DMA channel x:

1. Set the 32-bit source address in the register DMA_C0SrcAddr.
2. Set the 32-bit destination address in the register DMA_C0DstAddr.
3. Automatic address accumulation: You can set to enable/disable the automatic address accumulation mode by configuring the SI (source) and DI (destination) in the register DMA_C0Control. When it is set to 1, this mode is enabled.
4. You can set the width of transferred data by configuring the SWidth (source) and DWidth (destination) bits in the register DMA_C0Control, including single byte, double bytes and four bytes options.
5. You can set the Burst type by configuring the SBSIZE (source) and DBSIZE (destination) bits in the register DMA_C0Control, including INCR1, INCR4, INCR8, and INCR16 options.
6. It is worth noting that in the configured combination, the single burst of DMA must not exceed 16 bytes.
7. Range of data transfer length: 0-4095

7.3.3 Supported Peripherals

You can determine the peripherals supported by the current DMA by configuring the SrcPeripheral (source) and DstPeripheral (destination). The relationship between peripheral types and configuration values is shown as follows:

Value	DMA0
0	UART0_RX
1	UART0_TX
2	UART1_RX
3	UART1_TX
6	I2C0_RX
7	I2C0_TX
9	GPIO_TX
10	SPI0_RX
11	SPI0_TX
13	AUDIO_TX
14	I2C1_RX
15	I2C1_TX
16	I2S_RX
17	I2S_TX
20	DBI_TX
21	SOLO_RX
22	GPADC_RX
23	GPDAC_TX
24	FIO0_RX
25	FIO1_RX
26	FIO2_RX
27	FIO3_RX
28	FIO0_TX
29	FIO1_TX
30	FIO2_TX
31	FIO3_TX

Fig. 7.2: Peripheral Type Selection

Partial peripheral configuration examples:

UART uses DMA to transfer data

When UART sends data packets, DMA can greatly shorten the CPU's processing time, so that CPU resources will not be highly wasted, especially when UART sends and receives a large number of data packets (such as sending and receiving instructions frequently).

For example, the UART0 transfer is configured as follows:

1. Set the value of the SrcPeripheral bit in the register DMA_C0Config to 1. That is, set Source peripheral to UART0_TX
2. Set the value of the DstPeripheral in the register DMA_C0Config to 0. That is, set Destination peripheral to UART0_RX

I2C uses DMA to transfer data

Configuration process:

1. Set the value of the SrcPeripheral bit in the register DMA_C0Config to 7. That is, set Source peripheral to I2C0_TX
2. Set the value of the DstPeripheral in the register DMA_C0Config to 6. That is, set Destination peripheral to I2C0_RX

SPI uses DMA to transfer data

Configuration process:

1. Set the value of the SrcPeripheral bit in the register DMA_C0Config to 11. That is, set Source peripheral to SPI0_TX
2. Set the value of the DstPeripheral in the register DMA_C0Config to 10. That is, set Destination peripheral to SPI0_RX

ADC0/1 use DMA to transfer data

Configuration process:

1. Set the value of the SrcPeripheral bit in the register DMA_C0Config to 22/23. That is, set Source peripheral to GPADC0/GPADC1.

7.3.4 Linked List Mode

DMA supports the linked list working mode. During a DMA read or write operation, data can be filled in the next linked list. After the data transfer of the current linked list is completed, the start address of the next linked list can be obtained by reading the value of the register DMA_C0LLI, to directly transfer the data of the next linked list. This ensures continuous and uninterrupted work during DMA transfer, and improves the efficiency of CPU and DMA.

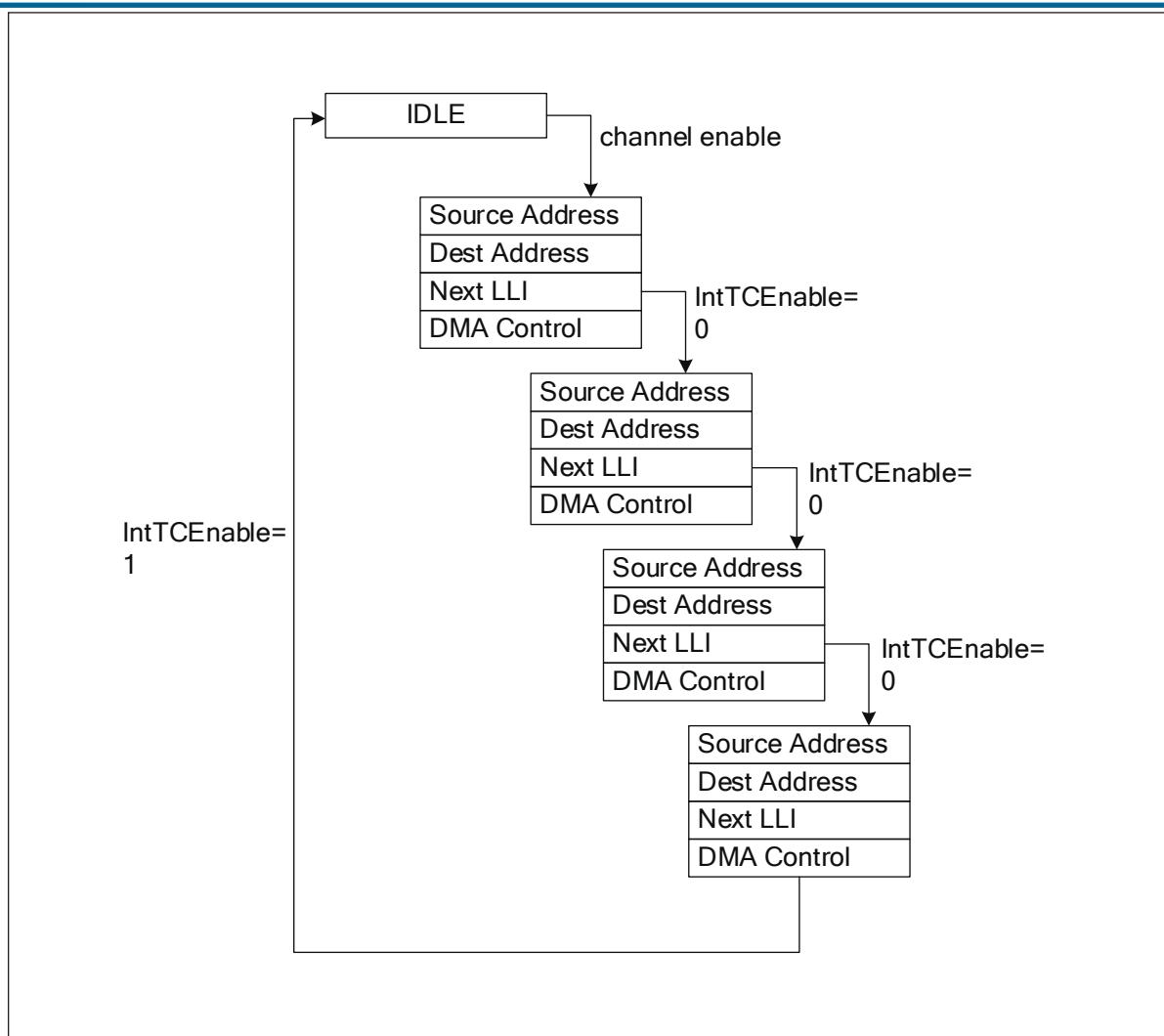


Fig. 7.3: LLI Framework

7.3.5 DMA Interrupt

- DMA_INT_TCOMPLETED
 - Data transfer complete interrupt: This interrupt will be generated when a data transfer is completed
 - DMA_INT_ERR
 - Data transfer error interrupt: This interrupt will be generated when an error occurs during data transfer

7.4 Transfer Mode

7.4.1 Memory to Memory

After this mode is enabled, DMA will transfer the data from the source address to the destination address based on the preset TransferSize. Upon transfer completed, the DMA controller will automatically return to the idle state and wait for the next transfer.

Configuration process:

1. Set the DMA_C0SrcAddr value of the register to the source memory address
2. Set the DMA_C0DstAddr value of the register to the destination memory address
3. Select a transfer mode, and set the value of the FlowCntrl bit in the register DMA_C0Config to 0, namely selecting the memorytomemory mode
4. Set the values of the corresponding bits in the register DMA_C0Control: the SI bit is set to 1, the DI bit is set to 0, the source address automatic accumulation mode is enabled; set the transfer width of the source and destination through SWidth and DWidth respectively; and set the burst type of the source and the destination through SBSIZE and DBSize respectively
5. Select a proper channel, enable DMA, and complete data transfer

7.4.2 Memory to Peripherals

In this mode, DMA will transfer the data from the source to the internal cache according to the preset TransferSize. The transfer will automatically pause when the cache space is insufficient and continue when there is enough cache space until the preset TransferSize is met.

Moreover, when the destination peripheral request is triggered, the target configuration will be burst to the destination address, and it will automatically return to the idle state until the preset TransferSize is met and wait for the next transfer.

Configuration process:

1. Set the DMA_C0SrcAddr value of the register to the source memory address
2. Set the DMA_C0DstAddr value of the register to the destination peripheral address
3. Select a transfer mode, and set the value of the FlowCntrl bit in the register DMA_C0Config to 1, namely selecting the Memorytoperipheral mode
4. Set the values of the corresponding bits in the register DMA_C0Control: the SI bit is set to 1, the DI bit is set to 0, the source address automatic accumulation mode is enabled; set the transfer width of the source and destination through SWidth and DWidth respectively; and set the burst type of the source and the destination through SBSIZE and DBSize respectively
5. Select a proper channel, enable DMA, and complete data transfer

7.4.3 Peripheral to Memory

In this mode, when the source peripheral request is triggered, the source configuration will be burst into the cache until the preset TransferSize is met. Moreover, when the internal cache is enough for one burst to the destination, DMA will automatically transfer the cached content to the destination address, automatically return to the idle state until the preset TransferSize is met, and wait for the next transfer.

Configuration process:

1. Set the DMA_C0SrcAddr value of the register to the source peripheral address
2. Set the DMA_C0DstAddr value of the register to the destination memory address
3. Select a transfer mode, and set the value of the FlowCntrl bit in the register DMA_C0Config to 2, namely selecting the Peripheraltomemory mode
4. Set the values of the corresponding bits in the register DMA_C0Control: Set the DI and SI bits to 1 to enable the automatic address accumulation mode; set the transfer width of the source and destination through SWidth and DWidth respectively; and set the burst type of the source and the destination through SBSIZE and DBSIZe respectively
5. Select a proper channel, enable DMA, and complete data transfer

7.5 Register description

Name	Description
DMA_IntStatus	
DMA_IntTCStatus	
DMA_IntTCClear	
DMA_IntErrorStatus	
DMA_IntErrClr	
DMA_RawIntTCStatus	
DMA_RawIntErrorStatus	
DMA_EnbldChns	
DMA_SoftBReq	
DMA_SoftSReq	
DMA_SoftLBReq	
DMA_SoftLSReq	
DMA_Config	

Name	Description
DMA_Sync	
DMA_C0SrcAddr	
DMA_C0DstAddr	
DMA_C0LLI	
DMA_C0Control	
DMA_C0Config	
DMA_C0RSVD	
DMA_C1SrcAddr	
DMA_C1DstAddr	
DMA_C1LLI	
DMA_C1Control	
DMA_C1Config	
DMA_C1RSVD	
DMA_C2SrcAddr	
DMA_C2DstAddr	
DMA_C2LLI	
DMA_C2Control	
DMA_C2Config	
DMA_C2RSVD	
DMA_C3SrcAddr	
DMA_C3DstAddr	
DMA_C3LLI	
DMA_C3Control	
DMA_C3Config	
DMA_C3RSVD	

7.5.1 DMA_IntStatus

Address: 0x2000c000

RSVD																							
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								

IntStatus

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntStatus	r	0	Status of the DMA interrupts after masking

7.5.2 DMA_IntTCStatus

Address: 0x2000c004

RSVD																							
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								

IntTCStatus

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntTCStatus	r	0	Interrupt terminal count request status

7.5.3 DMA_IntTCClear

Address: 0x2000c008

RSVD															
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

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD IntTCClear

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntTCClear	w	0	Terminal count request clear

7.5.4 DMA_IntErrorStatus

Address: 0x2000c00c

RSVD															
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

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD IntErrorStatus

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntErrorStatus	r	0	Interrupt error status

7.5.5 DMA_IntErrClr

Address: 0x2000c010

RSVD															
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

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD IntErrClr

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntErrClr	w	0	Interrupt error clear

7.5.6 DMA_RawIntTCStatus

Address: 0x2000c014

RSVD																	
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		

RawIntTCStatus

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	RawIntTCStatus	r	0	Status of the terminal count interrupt prior to masking

7.5.7 DMA_RawIntErrorStatus

Address: 0x2000c018

RSVD																	
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		

RawIntErrorStatus

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	RawIntErrorStatus	r	0	Status of the error interrupt prior to masking

7.5.8 DMA_EnbldChns

Address: 0x2000c01c

RSVD																							
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								

EnabledChannels

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	EnabledChannels	r	0	Channel enable status

7.5.9 DMA_SoftBReq

Address: 0x2000c020

SoftBReq

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

SoftBReq

Bits	Name	Type	Reset	Description
31:0	SoftBReq	r/w	0	Software burst request

7.5.10 DMA_SoftSReq

Address: 0x2000c024

SoftSReq

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

SoftSReq

Bits	Name	Type	Reset	Description
31:0	SoftSReq	r/w	0	Software single request

7.5.11 DMA_SoftLBReq

Address: 0x2000c028

SoftLBReq

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

SoftLBReq

Bits	Name	Type	Reset	Description
31:0	SoftLBReq	r/w	0	Software last burst request

7.5.12 DMA_SoftLSReq

Address: 0x2000c02c

SoftLSReq

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

SoftLSReq

Bits	Name	Type	Reset	Description
31:0	SoftLSReq	r/w	0	Software last single request

7.5.13 DMA_Config

Address: 0x2000c030

RSVD																
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	

Bits	Name	Type	Reset	Description
31:2	RSVD			
1	M	r/w	0	AHB Master endianness configuration: 0 = little-endian, 1 = big-endian
0	E	r/w	0	SMDMA Enable.

7.5.14 DMA_Sync

Address: 0x2000c034

DMA_Sync

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

DMA_Sync

Bits	Name	Type	Reset	Description
31:0	DMA_Sync	r/w	0	DMA synchronization logic for DMA request signals: 0 = enable, 1 = disable

7.5.15 DMA_C0SrcAddr

Address: 0x2000c100

SrcAddr

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

SrcAddr

Bits	Name	Type	Reset	Description
31:0	SrcAddr	r/w	0	DMA source address

7.5.16 DMA_C0DstAddr

Address: 0x2000c104

DstAddr

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

DstAddr

Bits	Name	Type	Reset	Description
31:0	DstAddr	r/w	0	DMA Destination address

7.5.17 DMA_C0LLI

Address: 0x2000c108

LLI

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

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	First linked list item. Bits [1:0] must be 0.

7.5.18 DMA_C0Control

Address: 0x2000c10c

1	Prot				DI	SI	fix_cnt				DWidth		RSVD	SWidth	dst_add_mode	DBSize
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	

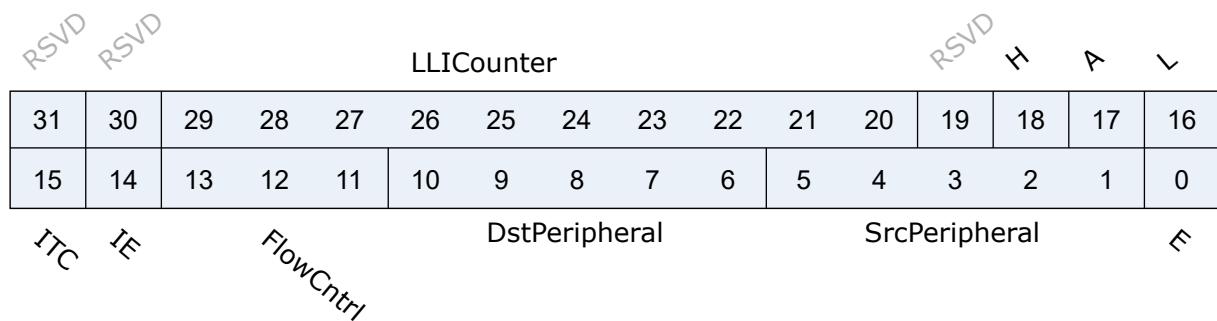
TransferSize

DBSize dst_min_mode SBSIZE

Bits	Name	Type	Reset	Description
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_-cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0 4095. Number of data transfers left to complete when the SMDMA is the flow controller.

7.5.19 DMA_C0Config

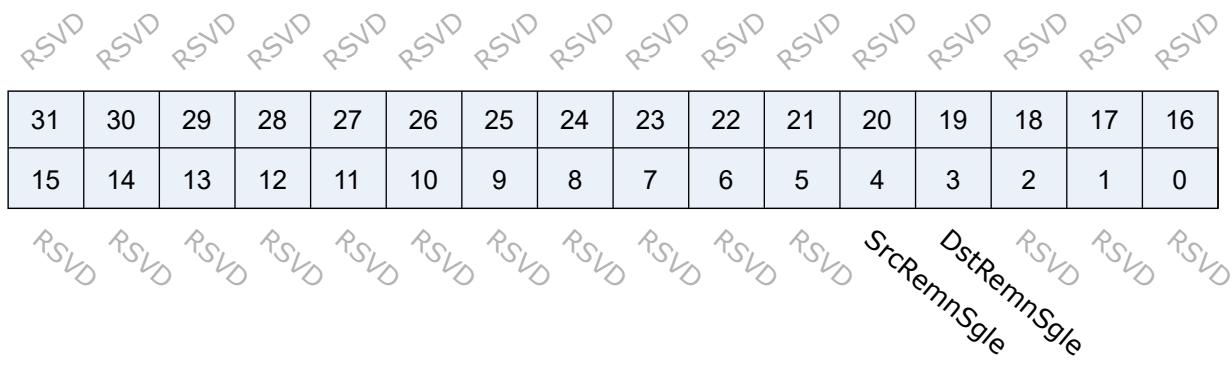
Address: 0x2000c110



Bits	Name	Type	Reset	Description
31:30	RSVD			
29:20	LLICounter	r	0	LLI counter. Increased 1 each LLI run. Cleared 0 when config Control.
19	RSVD			
18	H	r/w	0	Halt: 0 = enable DMA requests, 1 = ignore subsequent source DMA requests.
17	A	r	0	Active: 0 = no data in FIFO of the channel, 1 = FIFO of the channel has data.
16	L	r/w	0	Lock.
15	ITC	r/w	0	Terminal count interrupt mask.
14	IE	r/w	0	Interrupt error mask.
13:11	FlowCntrl	r/w	0	000: Memory-to-memory (DMA) 001: Memory-to-peripheral (DMA) 010: Peripheral-to-memory (DMA) 011: Source peripheral-to-Destination peripheral (DMA) 100: Source peripheral-to-Destination peripheral (Destination peripheral) 101: Memory-to-peripheral (peripheral) 110: Peripheral-to-memory (peripheral) 111: Source peripheral-to-Destination peripheral (Source peripheral)
10:6	DstPeripheral	r/w	0	Destination peripheral.
5:1	SrcPeripheral	r/w	0	Source peripheral.
0	E	r/w	0	Channel enable.

7.5.20 DMA_C0RSVD

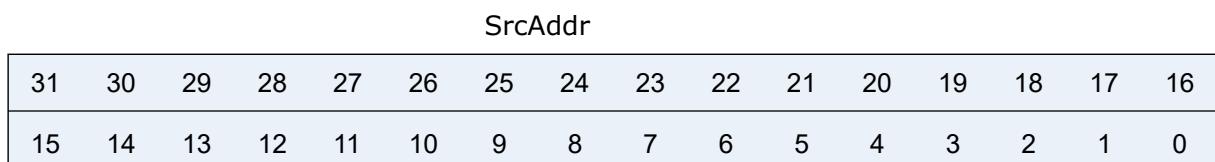
Address: 0x2000c11c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.21 DMA_C1SrcAddr

Address: 0x2000c200

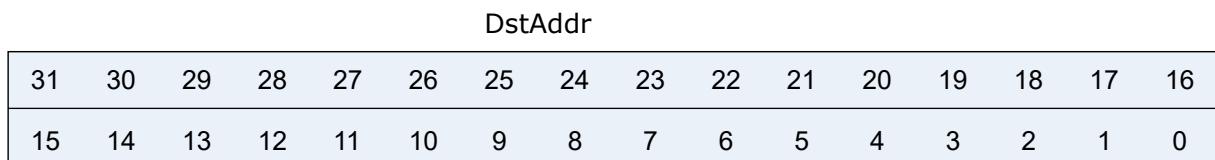


SrcAddr

Bits	Name	Type	Reset	Description
31:0	SrcAddr	r/w	0	

7.5.22 DMA_C1DstAddr

Address: 0x2000c204



DstAddr

Bits	Name	Type	Reset	Description
31:0	DstAddr	r/w	0	

7.5.23 DMA_C1LLI

Address: 0x2000c208

LLI

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

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	

7.5.24 DMA_C1Control

Address: 0x2000c20c

1	Prot	DI	SI	fix_cnt	DWidth	RSVD	SWidth	dst_add_mode	DBSize
31	30	29	28	27	26	25	24	23	22
15	14	13	12	11	10	9	8	7	6

TransferSize

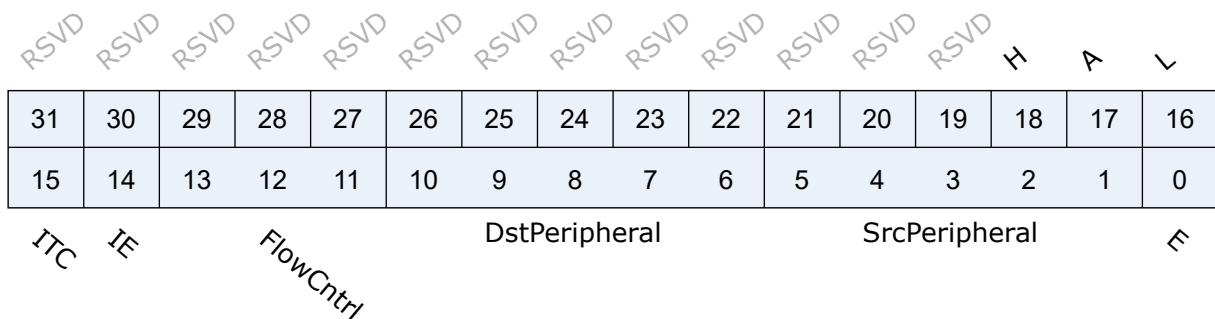
DBSize SBSIZE dst_min_mode

Bits	Name	Type	Reset	Description
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.

Bits	Name	Type	Reset	Description
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0 4095. Number of data transfers left to complete when the SMDMA is the flow controller.

7.5.25 DMA_C1Config

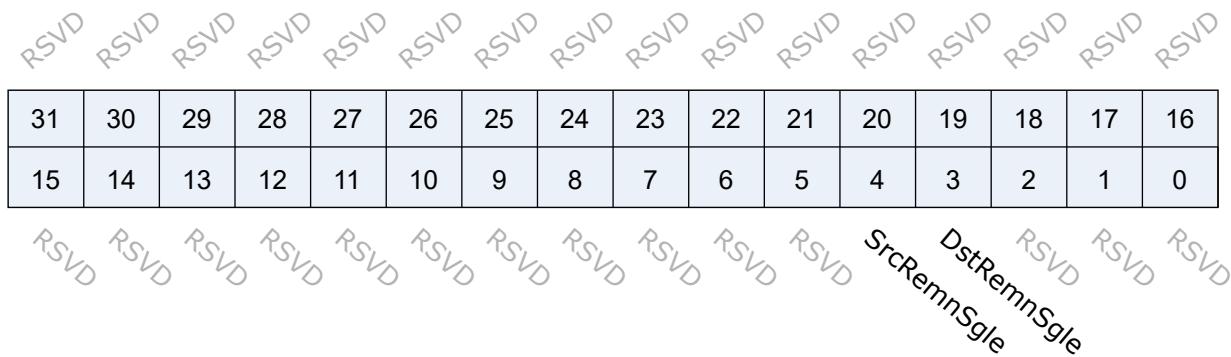
Address: 0x2000c210



Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

7.5.26 DMA_C1RSVD

Address: 0x2000c21c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.27 DMA_C2SrcAddr

Address: 0x2000c300

SrcAddr

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

SrcAddr

Bits	Name	Type	Reset	Description
31:0	SrcAddr	r/w	0	

7.5.28 DMA_C2DstAddr

Address: 0x2000c304

DstAddr

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

DstAddr

Bits	Name	Type	Reset	Description
31:0	DstAddr	r/w	0	

7.5.29 DMA_C2LLI

Address: 0x2000c308

LLI

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

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	

7.5.30 DMA_C2Control

Address: 0x2000c30c

1	Prot	DI	SI	fix_cnt	DWidth	RSVD	SWidth	dst_add_mode	DBSize
31	30	29	28	27	26	25	24	23	22
15	14	13	12	11	10	9	8	7	6
									TransferSize
									DBSize
									dst_min_mode

Bits	Name	Type	Reset	Description
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth

Bits	Name	Type	Reset	Description
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0 4095. Number of data transfers left to complete when the SMDMA is the flow controller.

7.5.31 DMA_C2Config

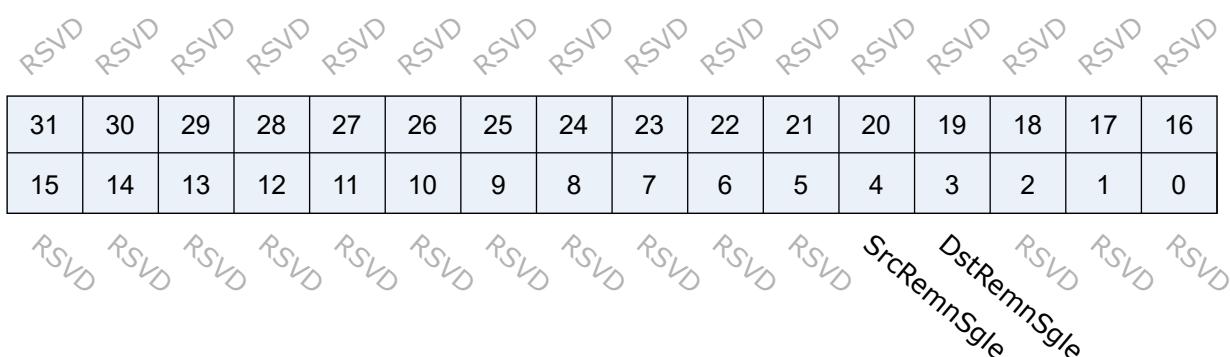
Address: 0x2000c310

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
<i>ITC</i>	<i>IF</i>	<i>FlowCntrl</i>				DstPeripheral				SrcPeripheral				<i>E</i>	

Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

7.5.32 DMA_C2RSVD

Address: 0x2000c31c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.33 DMA_C3SrcAddr

Address: 0x2000c400

SrcAddr

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

SrcAddr

Bits	Name	Type	Reset	Description
31:0	SrcAddr	r/w	0	

7.5.34 DMA_C3DstAddr

Address: 0x2000c404

DstAddr

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

DstAddr

Bits	Name	Type	Reset	Description
31:0	DstAddr	r/w	0	

7.5.35 DMA_C3LLI

Address: 0x2000c408

LLI

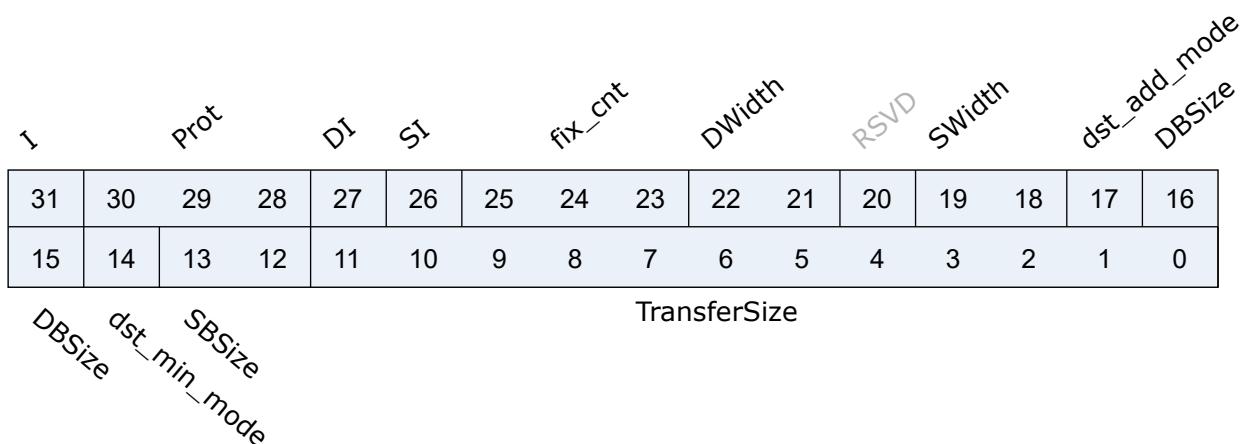
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	

7.5.36 DMA_C3Control

Address: 0x2000c40c

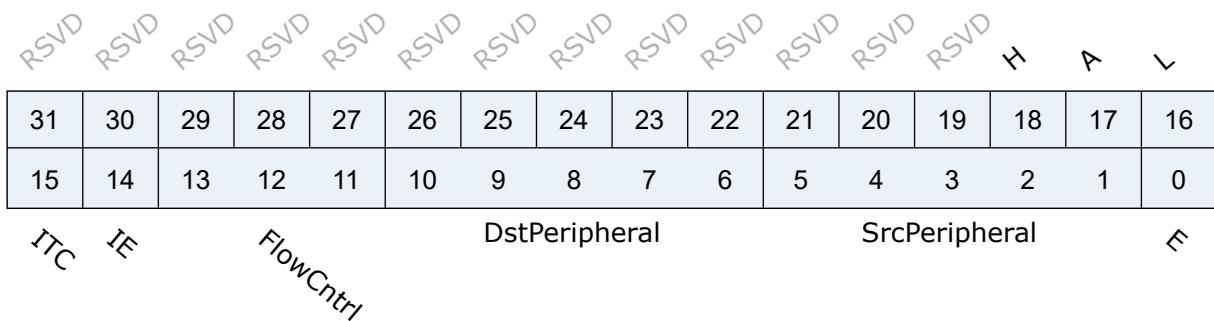


Bits	Name	Type	Reset	Description
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic

Bits	Name	Type	Reset	Description
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSIZE	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSIZE*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0 4095. Number of data transfers left to complete when the SMDMA is the flow controller.

7.5.37 DMA_C3Config

Address: 0x2000c410

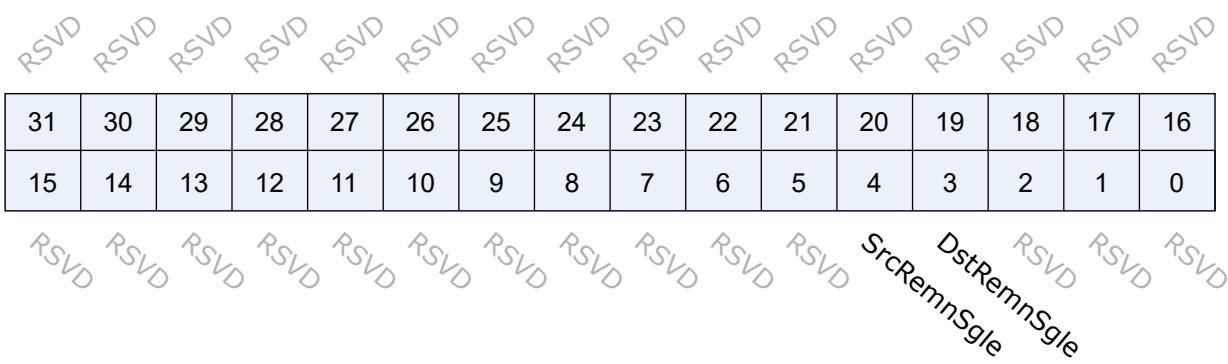


Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	

Bits	Name	Type	Reset	Description
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

7.5.38 DMA_C3RSVD

Address: 0x2000c41c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

8.1 Overview

Infrared Remote Control (IR) is a wireless and non-contact control technique that features strong anti-interference, reliable information transmission, low power consumption, and low cost. The IR radiating circuit uses the IR LED to emit the modulated infrared waves. The receiving circuit consists of infrared receiving diodes, triodes or silicon photocells, which convert the infrared light emitted by the infrared transmitter into corresponding electrical signals, and then send them to the post amplifier.

8.2 Features

- Receives data through the fixed NEC and RC5 protocols
- Receives data in any format by pulse width counting
- 64*2 bytes receive FIFO
- End of receiving interrupts

8.3 Functional Description

8.3.1 Fixed Protocol Based Receiving

IR receiving supports the fixed NEC and RC5 protocols.

- NEC Protocol

Logical ‘1’ and logical ‘0’ waveforms of the NEC protocol are shown as follows:

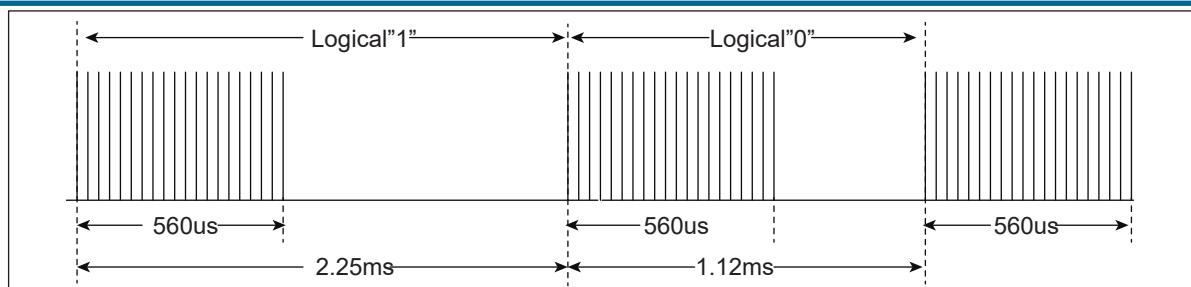


Fig. 8.1: NEC Logic Waveform

Logical '1' is 2.25 ms and the pulse time is 560 us. Logical '0' is 1.12 ms and the pulse time is 560 us. The format of the NEC protocol is shown as follows:

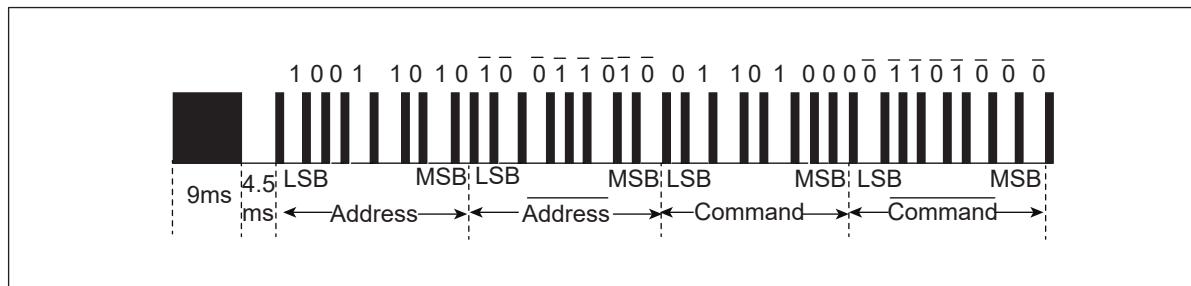


Fig. 8.2: NEC Protocol Waveform

The head pulse is a 9 ms high-level pulse and a 4.5 ms low-level pulse, followed by an 8-bit address code and its radix-minus-one complement (diminished radix complement), then an 8-bit command code and its radix complement, and the tail pulse is a 560 us high-level pulse and a 560 us low-level pulse.

- RC5 Protocol

Logical '1' and logical '0' waveforms of the RC5 protocol are shown as follows:

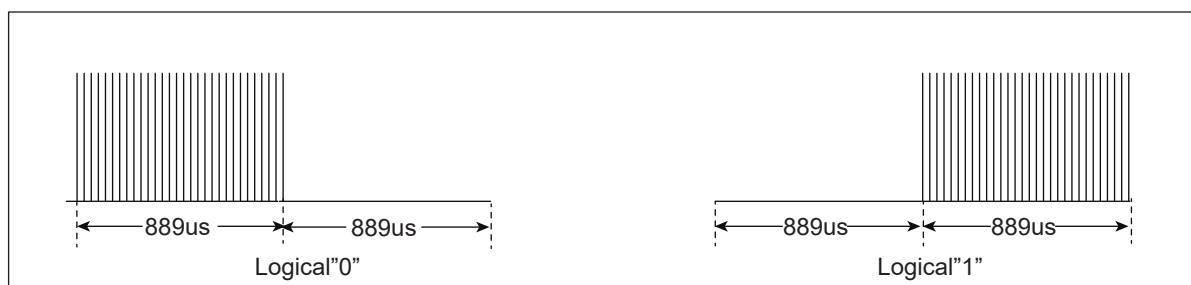


Fig. 8.3: RC5 Logic Waveform

The logical '1' is 1.778 ms, first the 889 us low level and then the 889 us high level. Logical '0' is opposite to the logical 1 waveform. The format of the RC5 protocol is shown as follows:

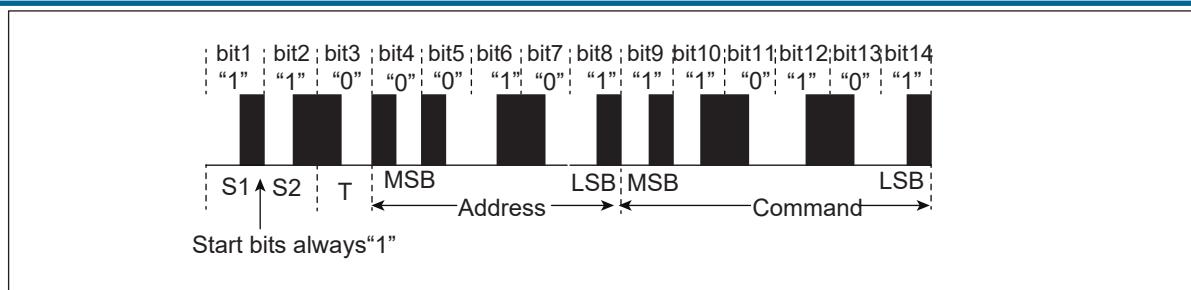


Fig. 8.4: RC5 Protocol Waveform

The first two bits are the Start bits, both logical ‘1’ . The third bit is the toggle bit, which toggles each time a key is released and pressed again. The next 5 bits are address bits and the next 6 bits are command bits.

It is worth noting that to improve the receiving sensitivity, the common infrared integrated receiver outputs a low level after receiving a high level, so you need to enable the receiving toggle function when using the IR receiving function.

8.3.2 Pulse width receiving

For data in any format other than the NEC and RC5 protocols, IR will count the duration of each high or low level in turn by its clock, and then store the data into the RX FIFO with a depth of 64 and a width of 2 bytes.

8.3.3 IR Interrupt

IR has a separate receiving end interrupt. During the receiving process, it will use the internal working clock to count the time that each level is maintained. Once the count reaches the set end threshold, a receiving end interrupt will be generated. The receive interrupt status and clear interrupt can be queried through the register IRRX_INT_STS.

8.4 Register description

Name	Description
irrx_config	
irrx_int_sts	
irrx_pw_config	
irrx_data_count	
irrx_data_word0	
irrx_data_word1	
ir_fifo_config_0	
ir_fifo_config_1	
ir_fifo_rdata	



8.4.1 irrx_config

Address: 0x40010240

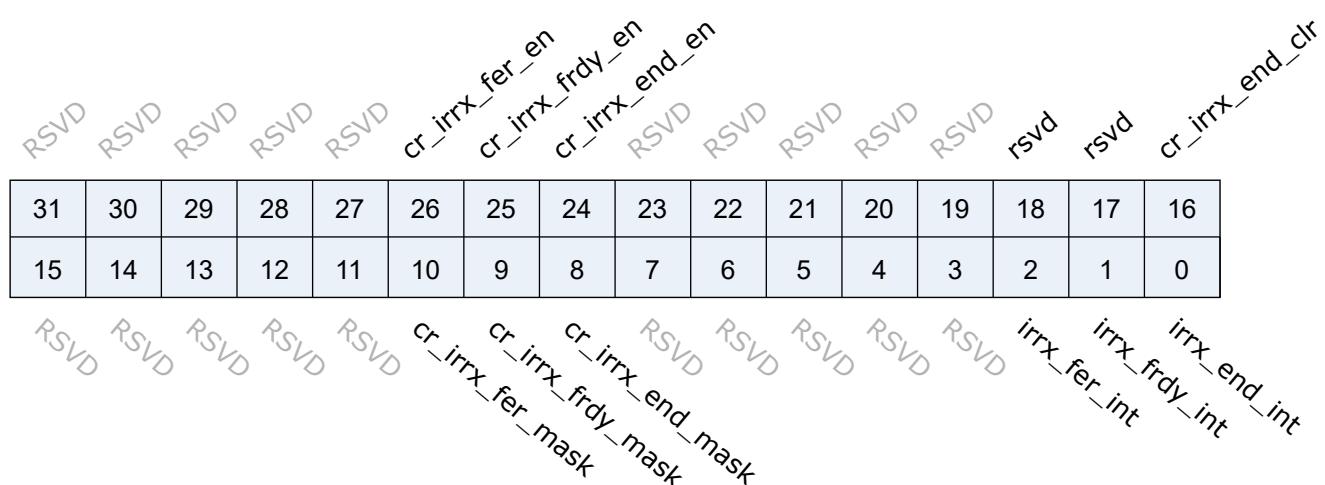
| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |

RSVD RSVD RSVD RSVD cr_irrx_deg_cnt RSVD RSVD RSVD cr_irrx_mode cr_irrx_deg_en cr_irrx_in_inv cr_irrx_en

Bits	Name	Type	Reset	Description
31:12	RSVD			
11:8	cr_irrx_deg_cnt	r/w	4'd0	De-glitch function cycle count
7:5	RSVD			
4	cr_irrx_deg_en	r/w	1'b0	Enable signal of IRRX input de-glitch function
3:2	cr_irrx_mode	r/w	2'd0	IRRX mode 0: NEC 1: RC5 2: SW pulse-width detection mode (SWM) 3: Reserved
1	cr_irrx_in_inv	r/w	1'b1	Input inverse signal
0	cr_irrx_en	r/w	1'b0	Enable signal of IRRX function Asserting this bit will trigger the transaction, and should be de-asserted after finish

8.4.2 irrx_int_sts

Address: 0x40010244



Bits	Name	Type	Reset	Description
31:27	RSVD			
26	cr_irrx_fer_en	r/w	1'b1	Interrupt enable of irrx_fer_int
25	cr_irrx_frdy_en	r/w	1'b1	Interrupt enable of irrx_frdy_int
24	cr_irrx_end_en	r/w	1'b1	Interrupt enable of irrx_end_int
23:19	RSVD			
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_irrx_end_clr	w1c	1'b0	Interrupt clear of irrx_end_int
15:11	RSVD			
10	cr_irrx_fer_mask	r/w	1'b1	Interrupt mask of irrx_fer_int
9	cr_irrx_frdy_mask	r/w	1'b1	Interrupt mask of irrx_frdy_int
8	cr_irrx_end_mask	r/w	1'b1	Interrupt mask of irrx_end_int
7:3	RSVD			
2	irrx_fer_int	r	1'b0	IRRX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
1	irrx_frdy_int	r	1'b0	IRRX FIFO ready ($rx_fifo_cnt > rx_fifo_th$) interrupt, auto-cleared when data is popped
0	irrx_end_int	r	1'b0	IRRX transfer end interrupt

8.4.3 irrx_pw_config

Address: 0x40010248

cr irrx end th

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

cr irrx data th

Bits	Name	Type	Reset	Description
31:16	cr_irrx_end_th	r/w	16'd8999	Pulse width threshold to trigger END condition
15:0	cr_irrx_data_th	r/w	16'd3399	Pulse width threshold for Logic0/1 detection (Don't care if SWM is enabled)

8.4.4 irrx data count

Address: 0x40010250

RSVD	RSVD																		
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				
RSVD	sts_irrx_data_cnt																		

Bits	Name	Type	Reset	Description
31:7	RSVD			
6:0	sts_irrx_data_cnt	r	7'd0	RX data bit count (pulse-width count for SWM)

8.4.5 irrx_data_word0

Address: 0x40010254

sts_irrx_data_word0

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

sts_irrx_data_word0

Bits	Name	Type	Reset	Description
31:0	sts_irrx_data_word0	r	32'h0	RX data word 0

8.4.6 irrx_data_word1

Address: 0x40010258

sts_irrx_data_word1

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

sts_irrx_data_word1

Bits	Name	Type	Reset	Description
31:0	sts_irrx_data_word1	r	32'h0	RX data word 1

8.4.7 ir_fifo_config_0

Address: 0x40010280

RSVD																
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	

RSVD RSVD

rx_fifo_overflow *rx_fifo_underflow* *rx_fifo_clr*

Bits	Name	Type	Reset	Description
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5:4	RSVD			
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2:0	RSVD			

8.4.8 ir_fifo_config_1

Address: 0x40010284

rx_fifo_th															
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
rx_fifo_cnt															
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

Bits	Name	Type	Reset	Description
31:30	RSVD			
29:24	rx_fifo_th	r/w	6'd0	RX FIFO threshold, irrx_frdy_int will not be asserted if rx_fifo_cnt is less than this value
23:15	RSVD			
14:8	rx_fifo_cnt	r	7'd0	RX FIFO available count
7:0	RSVD			

8.4.9 ir_fifo_rdata

Address: 0x4001028c

rx_fifo_rdata															
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
rx_fifo_rdata															
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	rx_fifo_rdata	r	16'h0	IRRX FIFO pulse width data for Software Mode

9.1 Overview

Serial Peripheral Interface (SPI) Bus is a synchronous serial communication interface specification for short-range communication. Devices communicate in the full duplex mode, which is a master-slave mode where one master controls one or more slaves.

SPI completes full-duplex communication with 4 signal lines, namely CS (chip select), SCLK (clock), MOSI (master output slave input), and MISO (master input slave output).

9.2 Features

- Can be used as SPI master or slave
- Both master and slave support 4 clock formats (CPOL, CPHA)
- Both master and slave support 1/2/3/4-byte transfer mode
- The sending and receiving channels each have a FIFO with a depth of 32 bytes
 - When the Frame is 32Bits, the depth of FIFO is 8
 - When the Frame is 24Bits, the depth of FIFO is 8
 - When the Frame is 16Bits, the depth of FIFO is 16
 - When the Frame is 8Bits, the depth of FIFO is 32
- Adjustable byte transfer sequence
- Flexible clock configuration, which supports up to 80M clock
- Configurable MSB/LSB transfer priority
- Receive ignore function: You can set to ignore the reception of data from the specified location

- Supports timeout mechanism in the slave mode
- Supports DMA transfer mode

9.3 Functional Description

9.3.1 Clock Control

Due to different clock phases and polarity settings, the SPI clock has four modes, which can be set by cr_spi_sclk_pol (CPOL) and cr_spi_sclk_ph (CPHA) in the register spi_config. CPOL determines the level of SCK clock signal when it is idle. If CPOL=0, the idle level is low, and if CPOL=1, the idle level is high. CPHA determines the sampling time. If CPHA=0, sampling is made at the first clock edge of each cycle, and if CPHA=1, that is made at the second clock edge of each cycle.

By setting the registers spi_prd_0 and spi_prd_1, you can also adjust the duration of the start and end levels of the clock, time of phase 0/1, and interval between frames of data. The settings of the four modes are shown as follows:

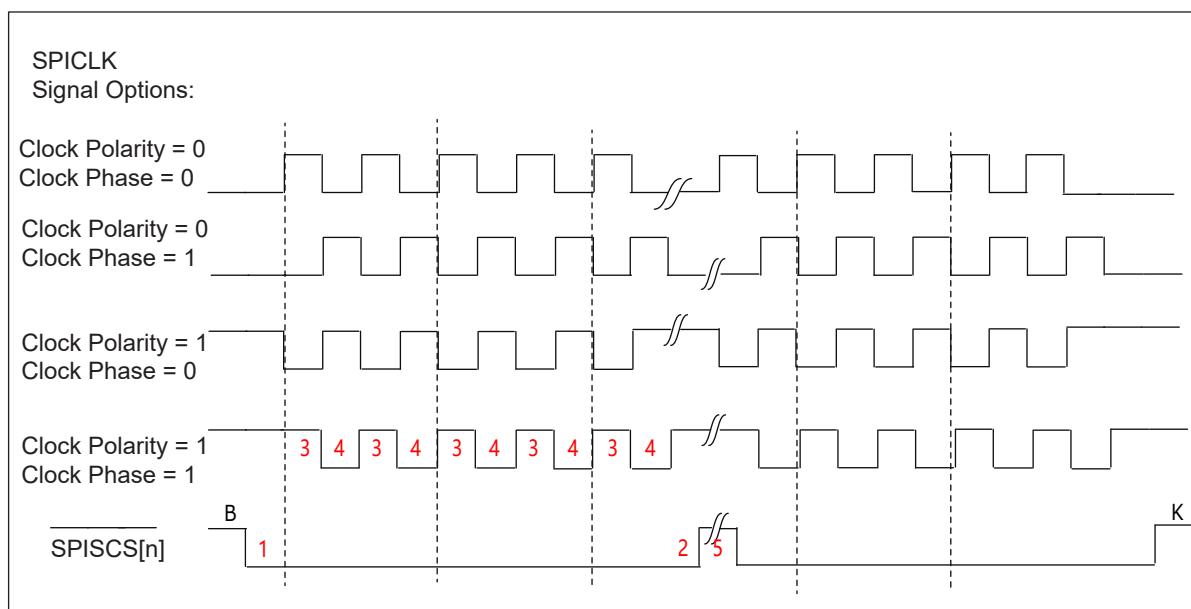


Fig. 9.1: SPI Timing

The meanings of numbers are as follows:

- “1” denotes the length of the START condition, which is configured by the cr_spi_prd_s in the register spi_prd_0.
- “2” denotes the length of the STOP condition, which is configured by the cr_spi_prd_p in the register spi_prd_0.
- “3” denotes the length of phase 0, which is configured by the cr_spi_prd_d_ph_0 in the register spi_prd_0.
- “4” denotes the length of phase 1, which is configured by the cr_spi_prd_d_ph_1 in the register spi_prd_0.
- “5” denotes the interval between frames of data, which is configured by the cr_spi_prd_i in the register spi_prd_1.

9.3.2 Master Continuous Transfer Mode

After this mode is enabled, the CS signal will not be released when the current data is sent and there are still available data in FIFO.

9.3.3 Master-Slave: Transfer and Receive Data

The same framesize shall be set for the master and slave for transferring and receiving data by configuring the cr_spi_frame_size in the register spi_config. When the master and slave agree to communicate at a 32 bits framesize, if the clk of the master does not meet 32 bits due to an exception in a frame of data, the following symptoms occur.

- The data sent by the master cannot be transferred to the RX FIFO of the slave. The slave cannot receive data from the master.
- When the slave sends data, it will skip this frame of data and continue to send the next frame of data when the master's clk is normal again.

9.3.4 Receive Ignore Function

When the start and end bits to be filtered out are set, SPI will discard the corresponding data segments in the received data, as shown below:

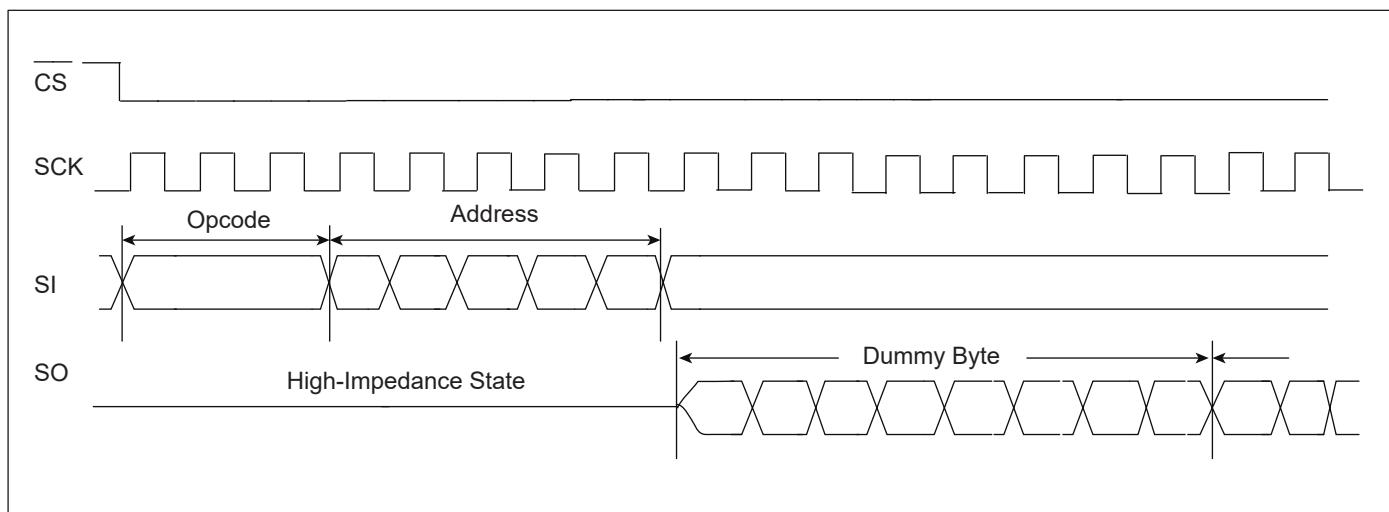


Fig. 9.2: SPI Ignore Waveform

You can enable this function by configuring the cr_spi_rxd_ignr_en in the register spi_config. The start bit of this function is set by configuring the cr_spi_rxd_ignr_s in the register spi_rxd_ignr. The end bit of this function is set by configuring the cr_spi_rxd_ignr_p in the register spi_rxd_ignr.

In the above figure, the start bit to be filtered is set to 0, and if the end bit is set to 7, Dummy Byte will be received; if the end bit is set to 15, Dummy Byte will be discarded.

9.3.5 Filtering Function

When this function is enabled and a threshold is set, SPI will filter the data less than or equal to the width threshold.

When this function is enabled by setting the cr_spi_deg_en in the register spi_config and the threshold is set by configuring the cr_spi_deg_cnt, SPI will filter out the data that cannot reach the width threshold. The data width shall be less than cr_spi_deg_cnt+1: As shown in the figure below, when the data width is 4, setting the cr_spi_deg_cnt to 4 can meet this condition. “Input” is the initial data and “output” is the filtered data.

Filtering logic process:

- “Tgl” is the exclusive OR result of input and output.
- “Deg_cnt” counts from 0, and the counting condition is that “tgl” is at the high level and “reached” is at the low level.
- “Reached” means whether the current deg_cnt count reaches the set cr_spi_deg_cnt, and it is at a high level once reached.
- When “reached” is at a high level, “input” is output to “output” .
- Note: user-defined condition for deg_cnt: “tgl” is at a high level and “reached” is at a low level. In other cases, “deg_cnt” will be cleared to 0.

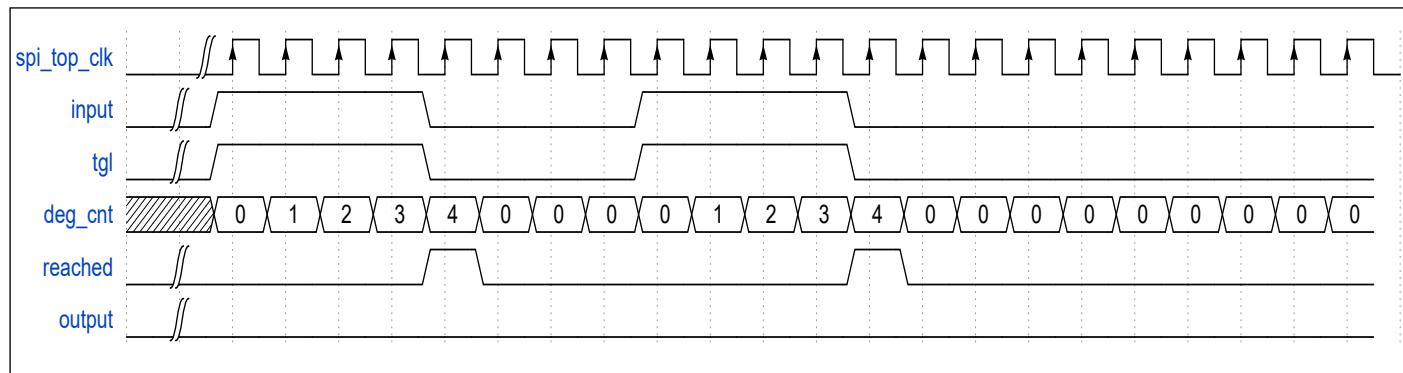


Fig. 9.3: SPI Filter Waveform

9.3.6 Configurable MSB/LSB Transfer

The configurable MSB/LSB transfer mode is limited to the priority transfer sequence of 8 bits in one byte, and the transfer sequence of bits in one byte is set by configuring the cr_spi_bit_inv bit in the register spi_config. 0 indicates MSB and 1 indicates LSB.

For example, for data transfer where the frame size is 24 bits, the data format is Data[23:0]=0x123456.

When MSB transfer is set, the transfer sequence is: 01010110 (binary, 1st byte: 0x56); 00110100 (binary, 2nd byte: 0x34); 00010010 (binary, 3rd byte: 0x12). When LSB transfer is set, the transfer sequence is: 01101010 (binary, 1st byte: 0x56); 00101100 (binary, 2nd byte: 0x34); 01001000 (binary, 3rd byte: 0x12).

9.3.7 Adjustable Byte Transfer Sequence

The adjustable byte transfer sequence is limited to the priority transfer sequence between different bytes in FIFO. The transfer sequence of bytes in FIFO is set by configuring the cr_spi_byte_inv bit in the register spi_config. 0 means sending LSB first, and 1 means sending MSB first.

For example, for data transfer where the frame size is 24 bits, the data format is Data[23:0]=0x123456.

When LSB transfer priority is set, the transfer sequence is 0x56 (1st byte: LSB); 0x34 (2nd byte: intermediate byte); 0x12 (3rd byte: MSB). When MSB transfer priority is set, the transfer sequence is 0x12 (3rd byte: MSB); 0x34 (2nd byte: intermediate byte); 0x56 (1st byte: LSB).

Adjustable byte transfer can be used in conjunction with configurable MSB/LSB transfer.

9.3.8 Slave Mode Timeout Mechanism

When a timeout threshold is set, an interrupt will be triggered when SPI in the slave mode receives no clock signal after the threshold exceeds.

9.3.9 I/O Transfer Mode

The chip communication processor can perform FIFO padding and clearing operations in response to the interrupt from the FIFO. Each FIFO has a programmable FIFO trigger threshold to trigger an interrupt. When rx_fifo_cnt in the register spi_fifo_config_1 is greater than the trigger threshold of rx_fifo_th, an interrupt will be generated to send a signal to the chip communication processor to clear the RX FIFO. When rx_fifo_cnt in the register spi_fifo_config_1 is greater than rx_fifo_th, an interrupt will be generated to send a signal to the chip communication processor to re-pad the TX FIFO.

You can query the SPI status register to determine the sampled value in the FIFO and the FIFO status. The software must provide correct trigger thresholds for RX FIFO and TX FIFO, and prevent the overflow of RX FIFO and the underflow of TX FIFO.

9.3.10 DMA Transfer Mode

SPI supports the DMA transfer mode. To enable this mode, you must set the thresholds of TX FIFO and RX FIFO respectively. Setting spi_dma_tx_en in the register spi_fifo_config_0 to 1 can enable the DMA sending mode. Setting spi_dma_rx_en in the register spi_fifo_config_0 to 1 can enable the DMA receiving mode. When this mode is enabled, UART will check the TX/RX FIFO. Once the tx_fifo_cnt/rx_fifo_cnt in the register spi_fifo_config_1 is greater than tx_fifo_th/rx_fifo_th, a DMA request will be initiated, and DMA will transfer data into TX FIFO or remove data from RX FIFO as configured.

9.3.11 SPI Interrupt

SPI supports the following interrupt control modes:

- SPI end of transfer interrupt
- TX FIFO request interrupt
- RX FIFO request interrupt
- Slave mode transfer timeout interrupt
- Slave mode TX overload interrupt
- TX/RX FIFO overflow interrupt

In the master mode, the SPI end of transfer interrupt will be triggered when the transfer of each frame of data ends. In the slave mode, that interrupt is triggered when the CS signal is released.

The TX/RX FIFO request interrupt will be triggered when the FIFO available count value is greater than the preset threshold, and the interrupt flag will be cleared automatically when the condition is unmet.

The slave mode transfer timeout interrupt will be triggered when no clock signal is received in the slave mode after the threshold exceeds. If the TX/RX FIFO overflows or underflows, it will trigger the TX/RX FIFO overflow interrupt.

Slave mode TX overload interrupt will trigger when the TX is not ready to transmit in slave mode.

When the tx_fifo_clr/rx_fifo_clr bit in the FIFO clear register spi_fifo_config_0 is set to 1, the corresponding FIFO will be cleared and the overflow interrupt flag will be cleared automatically.

You can query the interrupt status through the register SPI_INT_STS and write 1 to the corresponding bit to clear the interrupt.

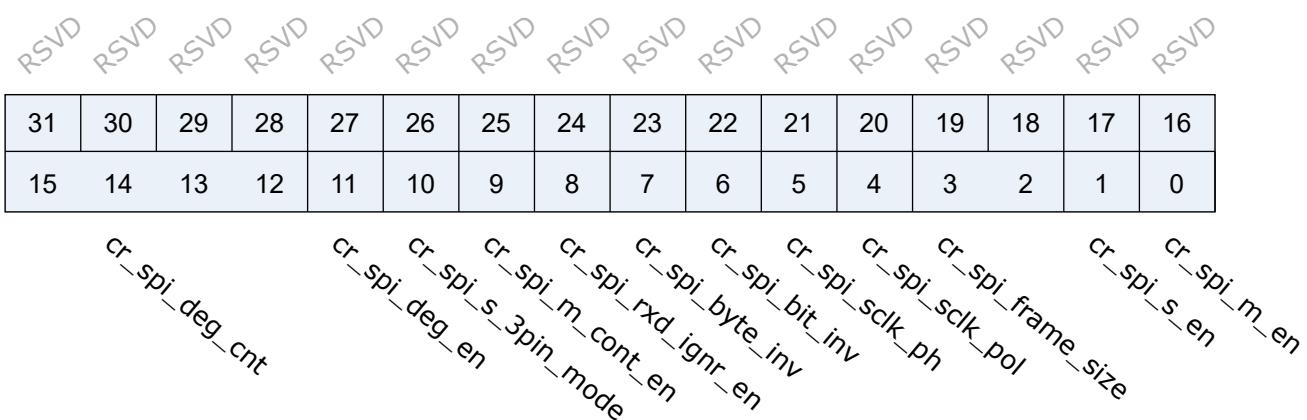
9.4 Register description

Name	Description
spi_config	
spi_int_sts	
spi_bus_busy	
spi_prd_0	
spi_prd_1	
spi_rxd_ignr	
spi_sto_value	
spi_fifo_config_0	

Name	Description
spi_fifo_config_1	
spi_fifo_wdata	
spi_fifo_rdata	
backup_io_en	

9.4.1 spi_config

Address: 0x40019000



Bits	Name	Type	Reset	Description
31:16	RSVD			
15:12	cr_spi_deg_cnt	r/w	4'd0	De-glitch function cycle count
11	cr_spi_deg_en	r/w	1'b0	Enable signal of all input de-glitch function
10	cr_spi_s_3pin_mode	r/w	1'b0	SPI slave 3-pin mode 1'b0: 4-pin mode (SS_n is enabled) 1'b1: 3-pin mode (SS_n is disabled / don't care)
9	cr_spi_m_cont_en	r/w	1'b0	Enable signal of master continuous transfer mode 1'b0: Disabled, SS_n will de-assert between each data frame 1'b1: Enabled, SS_n will stay asserted between each consecutive data frame if the next data is valid in the FIFO
8	cr_spi_rxd_ignr_en	r/w	1'b0	Enable signal of RX data ignore function

Bits	Name	Type	Reset	Description
7	cr_spi_byte_inv	r/w	1'b0	Byte-inverse signal for each FIFO entry data 0: Byte[0] is sent out first 1: Byte[3] is sent out first
6	cr_spi_bit_inv	r/w	1'b0	Bit-inverse signal for each data byte 0: Each byte is sent out MSB-first 1: Each byte is sent out LSB-first
5	cr_spi_sclk_ph	r/w	1'b0	SCLK clock phase inverse signal
4	cr_spi_sclk_pol	r/w	1'b0	SCLK polarity 0: SCLK output LOW at IDLE state 1: SCLK output HIGH at IDLE state
3:2	cr_spi_frame_size	r/w	2'd0	SPI frame size (also the valid width for each FIFO entry) 2'd0: 8-bit 2'd1: 16-bit 2'd2: 24-bit 2'd3: 32-bit
1	cr_spi_s_en	r/w	1'b0	Enable signal of SPI Slave function, Master and Slave should not be both enabled at the same time (This bit becomes don't-care if cr_spi_m_en is enabled)
0	cr_spi_m_en	r/w	1'b0	Enable signal of SPI Master function Asserting this bit will trigger the transaction, and should be de-asserted after finish

9.4.2 spi_int_sts

Address: 0x40019004

RSVD	RSVD	cr_spi_fer_en	cr_spi_txu_en	cr_spi_sto_en	cr_spi_rxf_en	cr_spi_txf_en	cr_spi_end_en	RSVD	RSVD	rsvd	cr_spi_txu_clr	cr_spi_sto_clr	rsvd	rsvd	cr_spi_end_clr
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
RSVD		cr_spi_fer_mask	cr_spi_txu_mask	cr_spi_sto_mask	cr_spi_rxf_mask	cr_spi_txf_mask	cr_spi_end_mask	RSVD		spi_fer_int	spi_txu_int	spi_sto_int	spi_rxf_int	spi_txf_int	spi_end_int

Bits	Name	Type	Reset	Description
31:30	RSVD			
29	cr_spi_fer_en	r/w	1'b1	Interrupt enable of spi_fer_int
28	cr_spi_txu_en	r/w	1'b1	Interrupt enable of spi_txu_int
27	cr_spi_sto_en	r/w	1'b1	Interrupt enable of spi_sto_int
26	cr_spi_rxf_en	r/w	1'b1	Interrupt enable of spi_rxv_int
25	cr_spi_txf_en	r/w	1'b1	Interrupt enable of spi_txe_int
24	cr_spi_end_en	r/w	1'b1	Interrupt enable of spi_end_int
23:22	RSVD			
21	rsvd	rsvd	1'b0	
20	cr_spi_txu_clr	w1c	1'b0	Interrupt clear of spi_txu_int
19	cr_spi_sto_clr	w1c	1'b0	Interrupt clear of spi_sto_int
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_spi_end_clr	w1c	1'b0	Interrupt clear of spi_end_int
15:14	RSVD			
13	cr_spi_fer_mask	r/w	1'b1	Interrupt mask of spi_fer_int
12	cr_spi_txu_mask	r/w	1'b1	Interrupt mask of spi_txu_int
11	cr_spi_sto_mask	r/w	1'b1	Interrupt mask of spi_sto_int
10	cr_spi_rxf_mask	r/w	1'b1	Interrupt mask of spi_rxv_int
9	cr_spi_txf_mask	r/w	1'b1	Interrupt mask of spi_txe_int
8	cr_spi_end_mask	r/w	1'b1	Interrupt mask of spi_end_int
7:6	RSVD			
5	spi_fer_int	r	1'b0	SPI TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
4	spi_txu_int	r	1'b0	SPI slave mode TX underrun error flag, triggered when TXD is not ready during transfer in slave mode
3	spi_sto_int	r	1'b0	SPI slave mode transfer time-out interrupt, triggered when SPI bus is idle for a given value
2	spi_rxf_int	r	1'b0	SPI RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto-cleared when data is popped
1	spi_txf_int	r	1'b1	SPI TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed

Bits	Name	Type	Reset	Description
0	spi_end_int	r	1'b0	SPI transfer end interrupt, shared by both master and slave mode Master mode: Triggered when the final frame is transferred Slave mode: Triggered when CS_n is de-asserted

9.4.3 spi_bus_busy

Address: 0x40019008

RSVD	sts_spi_bus_busy															
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	

9.4.4 spi_prd_0

Address: 0x40019010

cr_spi_prd_d_ph_1								cr_spi_prd_d_ph_0							
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

cr_spi_prd_p

cr_spi_prd_s

Bits	Name	Type	Reset	Description
31:24	cr_spi_prd_d_ph_1	r/w	8'd15	Length of DATA phase 1 (please refer to "Timing" tab)
23:16	cr_spi_prd_d_ph_0	r/w	8'd15	Length of DATA phase 0 (please refer to "Timing" tab)
15:8	cr_spi_prd_p	r/w	8'd15	Length of STOP condition (please refer to "Timing" tab)

Bits	Name	Type	Reset	Description
7:0	cr_spi_prd_s	r/w	8'd15	Length of START condition (please refer to "Timing" tab)

9.4.5 spi_prd_1

Address: 0x40019014

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

RSVD RSVD

cr_spi_prd_i

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	cr_spi_prd_i	r/w	8'd15	Length of INTERVAL between frame (please refer to "Timing" tab)

9.4.6 spi_rxd_ignr

Address: 0x40019018

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

RSVD RSVD

cr_spi_rxd_ignr_s

RSVD RSVD

cr_spi_rxd_ignr_p

Bits	Name	Type	Reset	Description
31:21	RSVD			
20:16	cr_spi_rxd_ignr_s	r/w	5'd0	Starting point of RX data ignore function
15:5	RSVD			
4:0	cr_spi_rxd_ignr_p	r/w	5'd0	Stopping point of RX data ignore function

9.4.7 spi_sto_value

Address: 0x4001901c

RSVD															
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

cr_spi_sto_value

Bits	Name	Type	Reset	Description
31:12	RSVD			
11:0	cr_spi_sto_value	r/w	12'hFFF	Time-out value for spi_sto_int triggering

9.4.8 spi_fifo_config_0

Address: 0x40019080

RSVD															
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

rx_fifo_underflow *rx_fifo_overflow* *tx_fifo_underflow* *tx_fifo_overflow* *tx_fifo_clr* *spi_dma_tx_en* *spi_dma_rx_en*

Bits	Name	Type	Reset	Description
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	spi_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface

Bits	Name	Type	Reset	Description
0	spi_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

9.4.9 spi_fifo_config_1

Address: 0x40019084

spi_fifo_config_1															
RSVD				rx_fifo_th				RSVD				tx_fifo_th			
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
rx_fifo_cnt								tx_fifo_cnt							

Bits	Name	Type	Reset	Description
31:29	RSVD			
28:24	rx_fifo_th	r/w	5'd0	RX FIFO threshold, dma_rx_req will not be asserted if rx_fifo_cnt is less than this value
23:21	RSVD			
20:16	tx_fifo_th	r/w	5'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:14	RSVD			
13:8	rx_fifo_cnt	r	6'd0	RX FIFO available count (unit: byte)
7:6	RSVD			
5:0	tx_fifo_cnt	r	6'd32	TX FIFO available count (unit: byte)

9.4.10 spi_fifo_wdata

Address: 0x40019088

spi_fifo_wdata															
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
spi_fifo_wdata															

Bits	Name	Type	Reset	Description
31:0	spi_fifo_wdata	w	x	<p>TX FIFO write data port</p> <p>Note: Partial valid if cr_spi_frame_size is set to different value:</p> <ul style="list-style-type: none"> 2'd0 (8-bit frame): Only [7:0] are valid and [31:8] are don't-care 2'd1 (16-bit frame): Only [15:0] are valid and [31:16] are don't-care 2'd2 (24-bit frame): Only [23:0] are valid and [31:24] are don't-care 2'd3 (32-bit frame): Entire [31:0] are valid

9.4.11 spi_fifo_rdata

Address: 0x4001908c

spi_fifo_rdata

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

spi_fifo_rdata

Bits	Name	Type	Reset	Description
31:0	spi_fifo_rdata	r	32'h0	<p>RX FIFO read data port</p> <p>Note: Partial valid if cr_spi_frame_size is set to different value:</p> <ul style="list-style-type: none"> 2'd0 (8-bit frame): Only [7:0] are valid and [31:8] are all 0s 2'd1 (16-bit frame): Only [15:0] are valid and [31:16] are all 0s 2'd2 (24-bit frame): Only [23:0] are valid and [31:24] are all 0s 2'd3 (32-bit frame): Entire [31:0] is valid



9.4.12 backup io en

Address: 0x400190fc

Bits	Name	Type	Reset	Description
31:1	RSVD			
0	backup_io_en	r/w	1'b0	

10

UART

10.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART) provides a flexible way to exchange full-duplex data with external devices. BL616/BL618 is provided with 2 UARTs, which can be used together with DMA to achieve efficient data communication.

10.2 Features

- Full-duplex asynchronous communication
- Optional data bit length: 5/6/7/8-bit
- Optional stop bit length: 0.5/1/1.5/2-bit
- Supports odd/even/none check bit
- Error-detectable start bit
- Abundant interrupt control modes
- Hardware flow control (RTS/CTS)
- Convenient baud rate programming
- Configurable MSB/LSB transfer priority
- Automatic baud rate detection of ordinary/fixed characters
- 32-byte TX/RX FIFO
- Supports DMA transfer mode
- Supports baud rate of 10 Mbps and below
- Supports the LIN bus protocol

- Supports the RS485 mode
- Optional clock sources: 160M/BCLK/XCLK
- Support filter function

10.3 Functional Description

10.3.1 Data Formats

The normal UART communication data consists of start bits, data bits, parity check bits, and stop bits. The UART of xx supports configurable data bits, parity check bits, and stop bits, which are set in registers utx_config and urx_config. The waveform of a frame of data is shown as follows:

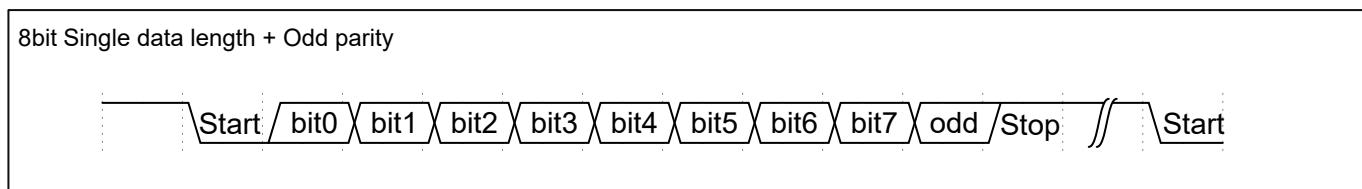


Fig. 10.1: UART Data Format

The start bit of the data frame occupies 1 bit, and the stop bit can be 0.5/1/1.5/2-bit wide by configuring the cr_utx_bit_cnt_p bit in the register utx_config. The start bit is at a low level and the stop bit is at a high level. The data bit width can be set to 5/6/7/8-bit by the cr_utx_bit_cnt_d bit in the register utx_config.

When the cr_utx_prt_en bit in the register utx_config and the cr_urx_prt_en bit in the register urx_config are set, the data frame adds a parity check bit after the data. The cr_utx_prt_sel bit in the register utx_config and the cr_urx_prt_sel bit in the register urx_config are used to select odd or even parity check. When the receiver detects the check bit error of the input data, it will generate the check error interrupt. However, the received data will still be stored into the FIFO.

Calculation method of odd parity check: If there is an odd number of “1” in the current data bit, the odd parity check bit is set to 0. Otherwise, it is set to 1.

Calculation method of even parity check: If there is an odd number of “1” in the current data bit, the even parity check bit is set to 1. Otherwise, it is set to 0.

10.3.2 Basic Architecture

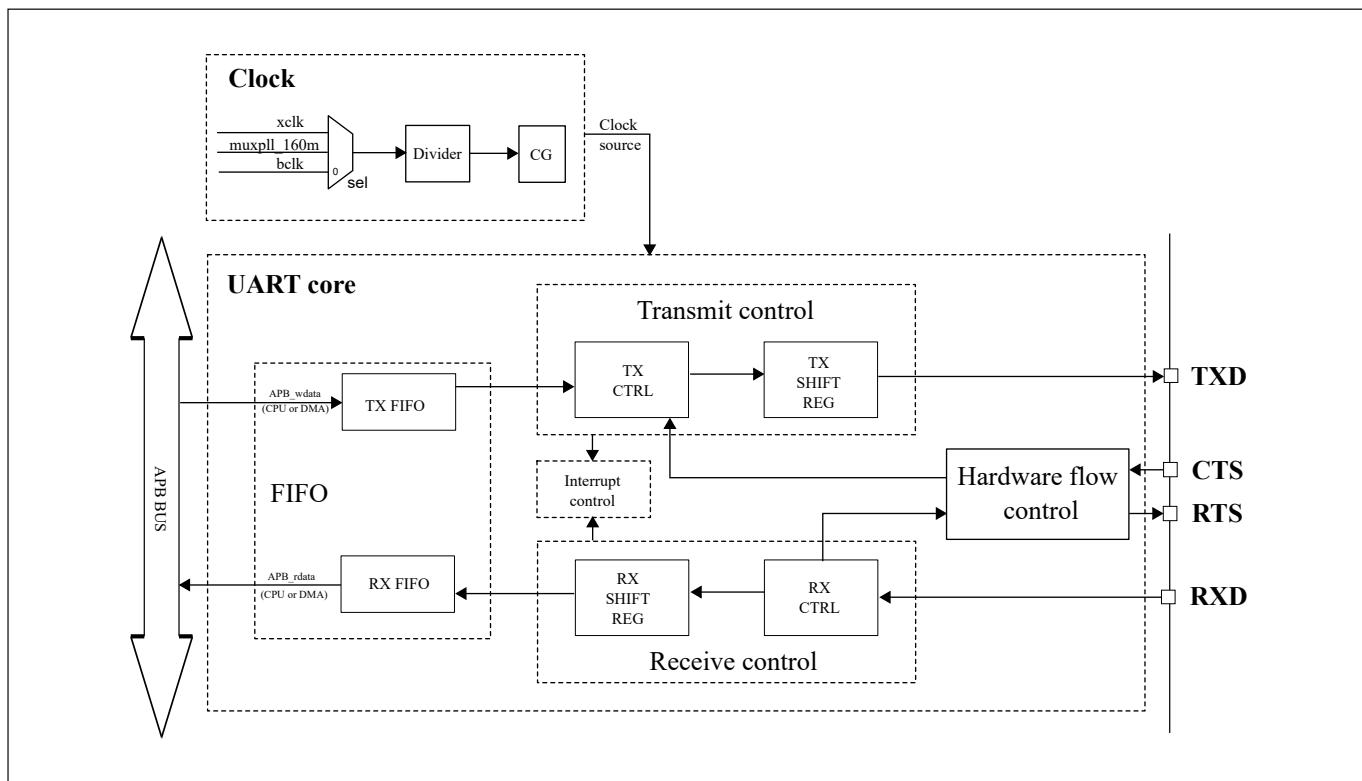


Fig. 10.2: UART Architecture

The UART has 3 clock sources: XCLK, 160MHz CLK and BCLK. The frequency divider in the clock is used to divide the frequency of the clock source and then generate the clock signal to drive the UART module.

The UART controller is divided into two functional blocks: transmitter and receiver.

10.3.3 Transmitter

The transmitter contains a 32-byte TX FIFO to store the data to be sent. When the transmission enable (TxEn) bit is set, the data stored in FIFO will be output from the TX pin. Software can transfer data into TX FIFO through DMA or APB bus. Software can check the status of transmitter by querying the remaining free space count value of TX FIFO through tx_fifo_cnt in the register uart_fifo_config_1.

FreeRun mode of transmitter:

- If the FreeRun mode is disabled, transmission will be terminated and an interrupt will be generated when the sent bytes reach the specified length. Before next transmission, you need to re-disable and enable the TxEn bit.
- If the FreeRun mode is enabled, the transmitter will send when there is data in the TX FIFO, and will not stop working because the sent bytes reach the specified length.

10.3.4 Receiver

The receiver contains a 32-byte RX FIFO to store the received data. Software can check the status of receiver by querying the available data count value of RX FIFO through rx_fifo_cnt in the register uart_fifo_config_1. The low 8 bits of the register URX_RTO_TIMER are used to set a receiving timeout threshold, which will trigger an interrupt when the receiver fails to receive data beyond the threshold. The cr_urx_deg_en and cr_urx_deg_cnt in the register urx_config are used to enable the deburring function and set the threshold, which control the filtering part before sampling by UART. UART will filter out the burrs whose width is lower than the threshold in the waveform and then send it to sampling.

10.3.5 Baud Rate Setting

$$\text{Baudrate} = \frac{\text{UART_clk}}{\text{uart_prd} + 1}$$

The user can set the baud rate of RX and TX separately. Take TX as an example: the value of uart_prd is the value of the lower 16 bits cr_utx_bit_prd of the register UART_BIT_PRD. Since the maximum value of the 16-bit bit width coefficient is 65535, the minimum baud rate supported by UART is : UART_clk/65536.

Before sampling the data, UART will filter the data to remove the burrs in the waveform. Then, the data will be sampled at the intermediate value of the 16-bit width factor, so that the sampling time can be adjusted based on baud rates, to ensure that the intermediate value is always sampled, providing much higher flexibility and accuracy. The sampling process is shown as follows:

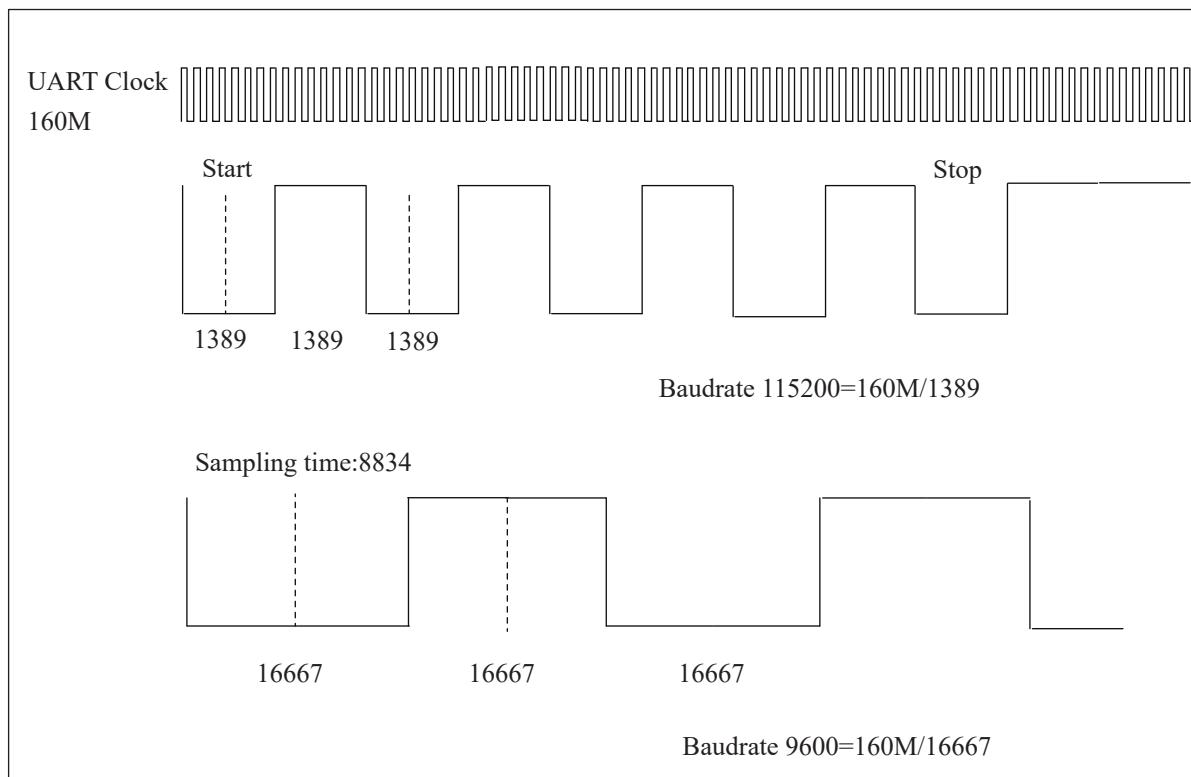


Fig. 10.3: UART Sampling Waveform

10.3.6 Filtering

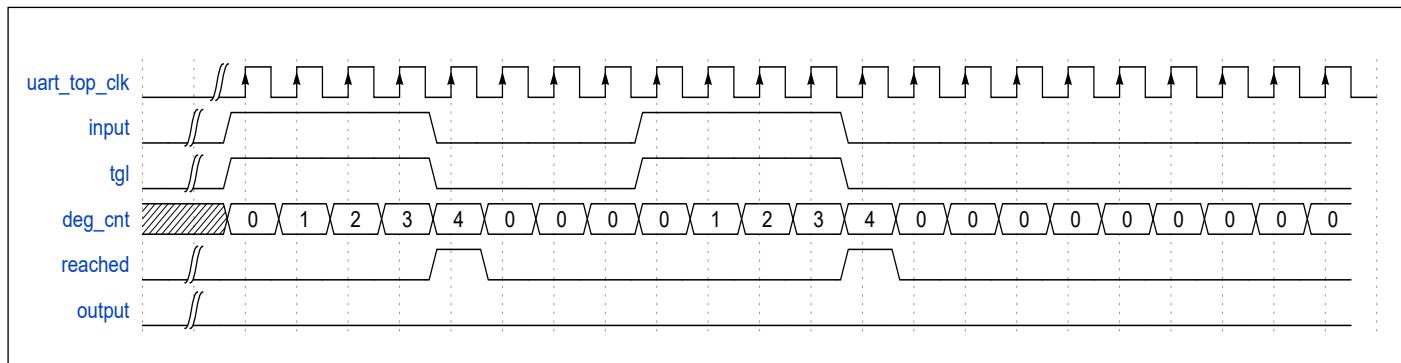


Fig. 10.4: UART Filter Waveform

When this function is enabled by configuring `cr_urx_deg_en` and the threshold is set by configuring `cr_urx_deg_cnt` in the register `urx_config`, UART will filter out the data that cannot meet the width threshold. As shown in the figure below, when the data width is 4, setting `cr_urx_deg_cnt` to 4 can meet this condition. “Input” is the initial data and “output” is the filtered data.

Filtering logic process:

- “Tgl” is the exclusive OR result of input and output.
- “Deg_cnt” counts from 0, and the counting condition is that “tgl” is at the high level and “reached” is at the low level.
- If the count value of `deg_cnt` reaches the value set by `cr_urx_deg_cnt`, `reached` is high level.
- When “reached” is at a high level, “input” is output to “output” .
- Note: user-defined condition for `deg_cnt`: “tgl” is at a high level and “reached” is at a low level. In other cases, “`deg_cnt`” will be cleared to 0.

10.3.7 Automatic Baud Rate Detection

UART supports automatic baud rate detection, which includes the general mode and fixed character mode. The two modes will be enabled every time `cr_urx_abr_en` in the register `urx_config` is set.

Common mode

For any character data received, UART will count the number of clocks in the start bit width, which will then be written into the low 16 bits `sts_urx_abr_prd_start` in the register `STS_URX_ABR_PRD` and used to calculate the baud rate. So the correct baud rate can be obtained when the first received data bit is 1, such as ‘0x01’ under LSBFIRST.

Fixed character (square wave) mode

In this mode, after the UART module counts the number of clocks in the bit width, it will continue to count the number of clocks in the subsequent data bits and compare it with the start bit. The check is passed, otherwise the count value

is discarded. The allowable error can be set by setting the cr_urx_abr_pw_tol bit in the register urx_abr_pw_tol, and the unit is the clock source of the UART.

Therefore, only when the fixed character ‘0x55’ / ‘0xD5’ under LSB-FIRST or ‘0xAA’ / ‘0xAB’ under MSB-FIRST is received, the UART module will write the clock count value in the starting bit width into the high 16-bit sts_urx_abr_prd_0x55 of the register STS_URX_ABR_PRD. As shown below:

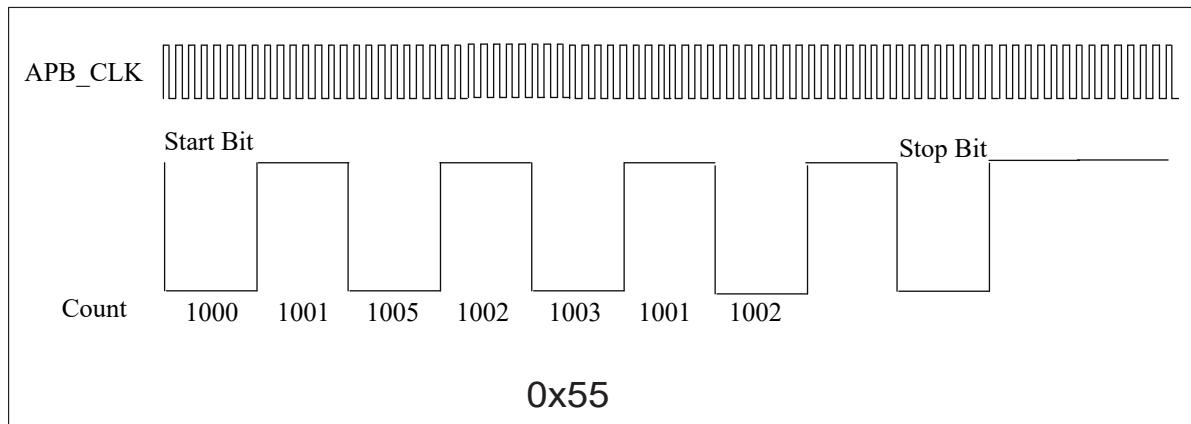


Fig. 10.5: Waveform of UART in fixed character mode

For an unknown baud rate, UART uses UART_CLK to count the width of the start bit and gets a count value of 1000, and the width of the second bit is 1001. If the width fluctuates for not more than four UART_CLK from the previous bit, UART will continue to count the third bit and get a count value of 1005. If the difference between the start bit and the third bit exceeds 4, the detection fails and the data will be discarded. UART compares the width of the first six data bits with that of the start bit in turn.

Formula for calculating the detected baud rate:

$$\text{Baudrate} = \frac{\text{UART_clk}}{\text{Count} + 1}$$

10.3.8 Hardware Flow Control

UART supports hardware flow control in CTS/RTS mode to prevent data in FIFO from being lost due to too late processing. Hardware flow control connection is shown as follows:

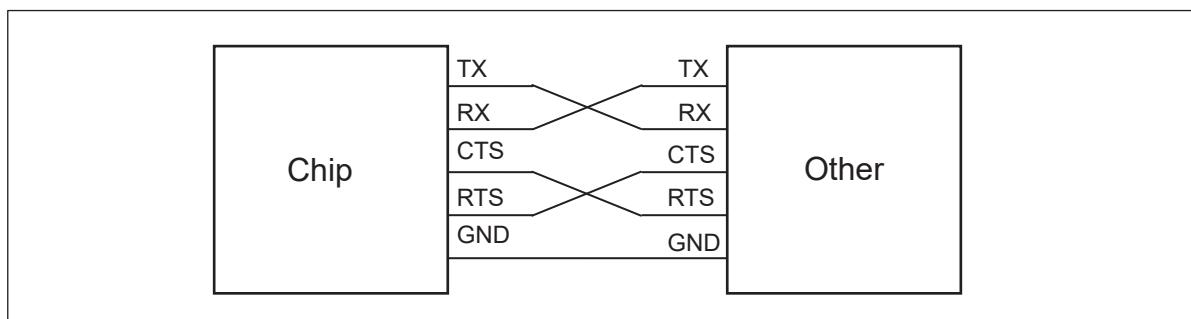


Fig. 10.6: UART hardware flow control

Require To Send (RTS) is an output signal, which indicates whether the chip is ready to receive data from the other side. This is valid at a low level denoting that the chip can receive data.

Clear To Send (CTS) is an input signal, which determines whether the chip can send data to the other side. This is valid at a low level denoting that the chip can send data to the other side.

When the hardware flow control function is enabled, the low level of chip's RTS indicates requesting the other side to send data, and the high level of that indicates informing the other side to stop sending data. When the chip detects that CTS goes high, TX will stop sending data, and continue sending until CTS goes low. If CTS goes high or low at any time during communication, it does not affect the continuity of data sent by TX, and the other side also can receive continuous data.

Two ways for hardware flow control of the transmitter:

- Hardware control (the cr_urx_rts_sw_mode in the register uart_sw_mode is 0): RTS goes high when cr_urx_en in the register urx_config is not turned on or the RX FIFO is almost full (one byte left).
- Software control(the cr_urx_rts_sw_mode in the register uart_sw_mode is 1): The level of RTS can be changed by configuring cr_urx_rts_sw_val in the register uart_sw_mode.

10.3.9 DMA Transfer

UART supports DMA transfer. Using DMA transfer, the TX and RX FIFO thresholds need to be set respectively by tx_fifo_th and rx_fifo_th in register uart_fifo_config_1. When this mode is enabled, if tx_fifo_cnt in uart_fifo_config_1 is greater than tx_fifo_th, a DMA TX request will be triggered. After the DMA is configured, when the DMA receives the request, it will move the data from the memory to the TX FIFO according to the settings. If the rx_fifo_cnt in uart_fifo_config_1 is greater than rx_fifo_th, the DMA RX request will be triggered. After the DMA is configured, when the DMA receives the request, it will transfer the data of the RX FIFO to the memory according to the settings.

In order to ensure the correctness of the data transferred by the chip DMA TX Channel, the following conditions need to be met in the Channel configuration: $(\text{transferWidth} * \text{burstSize}) \leq (\text{tx_fifo_th} + 1)$.

In order to ensure the integrity of the data transferred by the chip DMA RX Channel, the following conditions need to be met in the Channel configuration: $(\text{transferWidth} * \text{burstSize}) = (\text{rx_fifo_th} + 1)$.

10.3.10 Support for LIN Bus

The protocol for the Local Interconnect Network (LIN) is based on the Volvo-Lite technology developed by the Volvo spin-out company—Volcano Communications Technology (VCT). LIN is a complementary protocol to CAN and SAE J1850, suitable for applications that have low requirement for time or require no precise fault tolerance (as LIN is not as reliable as CAN). LIN aims to be easy to use as a low-cost alternative to CAN. The vehicle parts where LIN can be used include window regulator, rearview mirror, wiper, and rain sensor.

UART supports the LIN bus mode. The LIN bus is under the master-slave mode, and data is always initiated by the master node. The frame (header) sent by the master node contains synchronization interval field, synchronization byte field, and identifier field. A typical LIN data transfer is shown as follows.

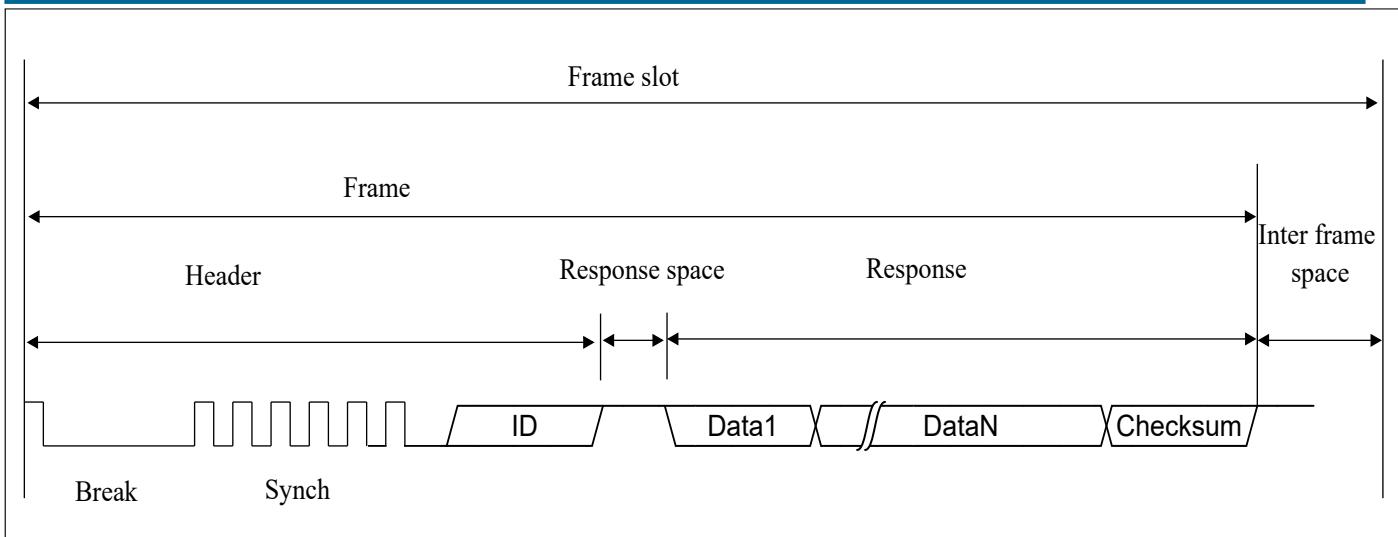


Fig. 10.7: A typical LIN frame

1. LIN Break Field

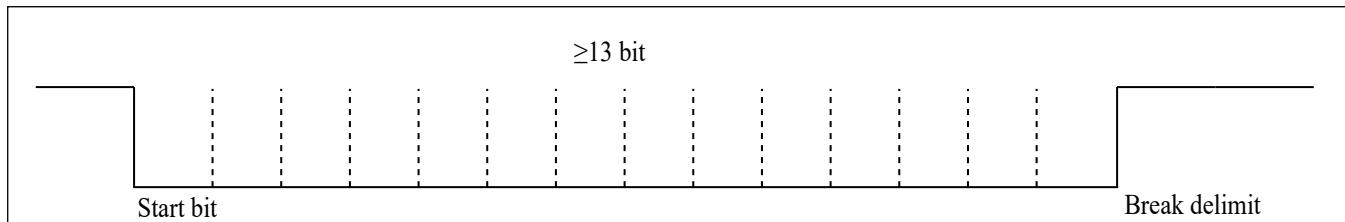


Fig. 10.8: Break Field of LIN

The synchronization interval field indicates the start of the message, with at least 13 dominant bits (including the start bit). The synchronization interval ends with an “interval separator”, which contains at least one recessive bit.

The length of the Break in the LIN frame can be set by `cr_utx_bit_cnt_b` in `utx_config`.

2. LIN Sync Field

A synchronization byte field is sent to determine the time between two falling edges, to determine the transmission rate used by the master node. The bit pattern is 0x55 (01010101, maximum number of falling edges).

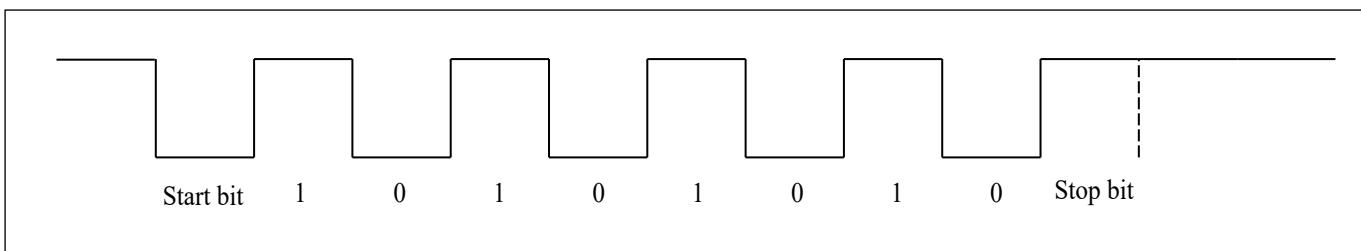


Fig. 10.9: Sync Field of LIN

3. LIN ID Field

The identifier field contains a 6-bit identifier and two parity check bits. The 6-bit identifier contains information about the sender and receiver, and the number of bytes required in the response. The parity check bit is calculated as follows: The check bit P0 is the result of logical OR operation among ID0, ID1, ID2, and ID4. The check bit P1 is the result of inversion after logical OR operation among ID1, ID3, ID4, and ID5.

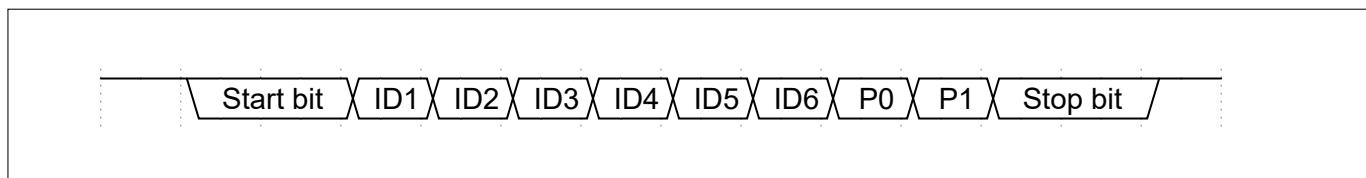


Fig. 10.10: ID Field of LIN

The slave node waits for the synchronization interval field, and then starts to synchronize between master and slave nodes through the synchronization byte field. Depending on the identifier sent by the master node, the slave node will receive, send or do not respond. The slave node that should send data sends the number of bytes requested by the master node, and then ends the transmission with a checksum field.

UART supports the LIN transfer mode. To enable this mode, you need to configure the cr_utx_bit_cnt_b by setting cr_utx_lin_en in the register utx_config so that the synchronization interval field consists of at least a 13-bit dominant level.

10.3.11 RS485 mode

UART supports the RS485 mode. After the cr_utx_rs485_en in the register UTX_RS485_CFG is set, UART can work in the RS485 mode. Then, UART can be connected to the RS485 bus through an external RS485 transceiver. In this mode, the RTS pin in the module performs the Dir function of transceiver. When UART has data to send, it will automatically control the RTS pin at a high level, so that the transceiver can send data to the bus. Contrarily, when UART has no data to send, it will automatically control the RTS at a low level, to keep the transceiver in the RX state.

UART supports the RS485 transfer mode. To enable this mode, you need to set cr_utx_rs485_pol and cr_utx_rs485_en in the register UTX_RS485_CFG.

10.3.12 UART Interrupt

UART supports the following interrupt control modes:

- TX end of transfer interrupt
- RX end interrupt
- TX FIFO request interrupt
- RX FIFO request interrupt
- RX timeout interrupt
- RX parity check error interrupt

- TX FIFO overflow interrupt
- RX FIFO overflow interrupt
- RX BCR interrupt
- LIN synchronization error interrupt
- Auto baud rate detection (universal mode) interrupt
- Auto baud rate detection (fixed characters mode) interrupt

10.3.13 TX/RX end of transfer interrupt

You can set a transfer length for TX and RX respectively by configuring the high 16 bits of the registers utx_config and urx_config. When the number of transferred bytes reaches this value, the corresponding TX/RX end of transfer interrupt will be triggered. While this interrupt is generated, TX stops working. To continue to use TX, you must re-initialize this module. Then, RX resumes to work. If the preset transfer length of TX is less than the data volume actually sent by TX, the other side can only receive the data equal to the transfer length, and the remaining data will be stored in TX FIFO. After this module is re-initialized, the data in TX FIFO can be sent out.

For TX, if the data continuously filled into the TX FIFO is greater than the set transmission length value, only the data of the set length value will be transmitted on the TX pin, and the excess data will be kept in the TX FIFO, and the TX will be re-enabled. After the function, the remaining data in the TX FIFO will be sent out.

For example: set the TX transmission length value to 64, enable the TX function, first fill 63 bytes into the TX FIFO, these 63 bytes will be transmitted on the pins, but no TX transmission completion interrupt is generated, and then the TX FIFO is sent to the TX FIFO. Fill in 1 byte, at this time, the transmission length reaches the transmission length value set by TX, a transmission completion interrupt will be generated, and the TX function will stop. Continue to fill 1 byte into the TX FIFO, you will find that there is no data transmission on the pin, the byte is still retained in the TX FIFO, and the TX function is turned off and re-enabled, and the byte is sent out on the pin.

For RX, if the data length sent by the other party exceeds the set transmission length, RX can continue to receive data after the RX transmission completion interrupt is generated.

For example: set the RX transmission length value to 16, the other party sends 32 bytes of data, RX will generate an RX transmission completion interrupt when it receives 16 bytes of data, and continue to receive the remaining 16 bytes of data, all saved in the RX FIFO.

10.3.14 TX/RX FIFO request interrupt

A TX FIFO request interrupt will be generated when tx_fifo_cnt in uart_fifo_config_1 is greater than tx_fifo_th. When the condition is not met, the interrupt flag will be cleared automatically.

A RX FIFO request interrupt will be generated when rx_fifo_cnt in uart_fifo_config_1 is greater than rx_fifo_th. When the condition is not met, the interrupt flag will be cleared automatically.

TX/RX supports multiple rounds of transmission/receiving, instead of reaching the value set by tx_fifo_th/rx_fifo_th at a time.

E.g:

1. Set tx_fifo_th/rx_fifo_th in register uart_fifo_config_1 to 16.
2. Set cr_utx_frm_en in register utx_config to enable free run mode.
3. Set cr_utx_frdy_mask/cr_urx_frdy_mask in register uart_int_mask to 0, and enable FIFO interrupt of TX/RX.
4. Set cr_utx_en/cr_urx_en in register utx_config/urx_config to enable TX/RX.
5. TX FIFO interrupt: TX will always enter the FIFO interrupt, when the chip sends 128 bytes, set cr_utx_frdy_mask to 1 to shield the interrupt. If you want to enter the TX FIFO interrupt again, set cr_utx_frdy_mask to 0.
6. RX FIFO interrupt: the other party first sends 15 bytes, no interrupt is generated, at this time, the value of rx_fifo_cnt is 15, and an interrupt is generated when 1 byte is sent again to reach the value set by rx_fifo_th. After the transmission is interrupted, the other party sends the data again, and the chip can receive the data.

10.3.15 RX timeout interrupt

The RX timeout interrupt generation condition is: after receiving data last time, the receiver will start timing, and the interrupt will be triggered when the timing value exceeds the timeout threshold and the next data has not been received. The time-out threshold value is in the unit of communication bit.

When the other party sends data to the chip, a timeout interrupt will be generated after the set timeout period is reached.

10.3.16 RX parity check error interrupt

The RX parity check error interrupt will be generated when a parity check error occurs. But it does not affect the RX, which still can correctly receive and analyze the data sent by the other side. When receiving data, RX takes the first 8 bits as data bits and ignores parity check bits, ensuring data consistency.

For example, you can enable parity check by setting cr_utx_prt_en/cr_urx_prt_en in the register utx_config/urx_config, and select the parity check type by setting cr_utx_prt_sel/cr_urx_prt_sel in the register utx_config/urx_config. When the other side sends data to the chip through odd/even parity check, the parity check of RX is disabled, but RX can receive correct data.

10.3.17 TX/RX FIFO overflow interrupt

If the TX/RX FIFO overflows or underflows, it will trigger the corresponding overflow interrupt. When the tx_fifo_clr and rx_fifo_clr bits in the FIFO clear bit register UART_FIFO_CONFIG_0 are set to 1, the corresponding FIFO will be cleared and the overflow interrupt flag will be cleared automatically. You can query the interrupt status through the register UART_INT_STS, and clear the interrupt by writing 1 to the corresponding bit in the register UART_INT_CLEAR.

10.3.18 RX BCR interrupt

A BCR interrupt will be generated when the data received by RX reaches the value set by cr_urx_bcr_value in the register urx_bcr_int_cfg.

The difference from RX END interrupt is that END interrupt is suitable for receiving data of known length, while BCR interrupt can be used to receive interrupts of unknown length. The trigger position of the END interrupt is controlled by cr_urx_len, and the counter will be cleared to 0 when an interrupt is triggered. The trigger position of BCR interrupt is controlled by cr_urx_bcr_value. When the interrupt is triggered, the counter will accumulate instead of being cleared to 0, but it can be cleared by software (cr_urx_bcr_clr). When the BCR interrupt is used together with the chained DMA, if you do not know how much data has been transferred by DMA, you can check it through “count” .

10.3.19 LIN synchronization error interrupt

When cr_utx_lin_en in utx_config is enabled, the LIN mode is enabled. Then, if the synchronization field of the LIN bus is not detected when data is received in this mode, the LIN synchronization error interrupt will be generated.

10.3.20 Auto baud rate detection (universal/fixed characters mode) interrupt

In the auto baud rate detection mode, when a baud rate is detected, the auto baud rate detection (universal/fixed characters mode) interrupt will be generated as configured.

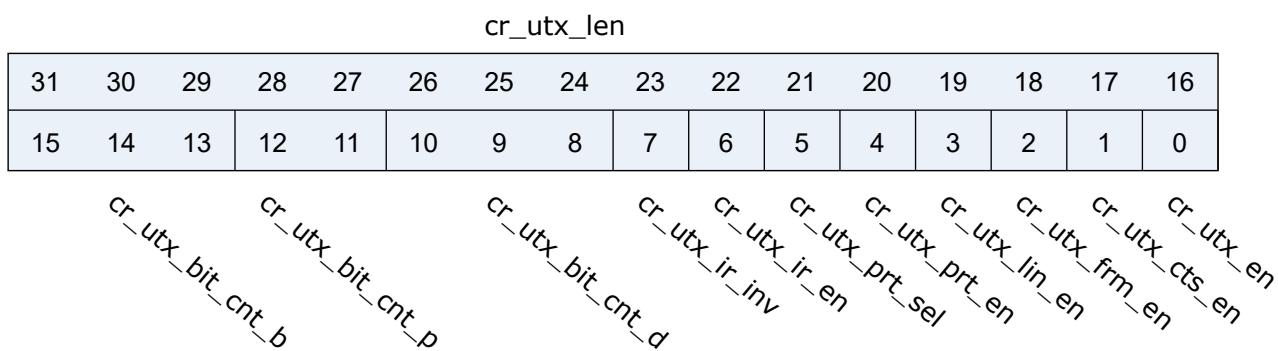
10.4 Register description

Name	Description
utx_config	TX configure
urx_config	RX configure
uart_bit_prd	Baud rate setting
data_config	LSB/MSB-first select
utx_ir_position	IR mode TX setting
urx_ir_position	IR mode RX setting
urx_rto_timer	Time-out value setting

Name	Description
uart_sw_mode	software mode
uart_int_sts	UART interrupt status
uart_int_mask	UART interrupt mask
uart_int_clear	UART interrupt clear
uart_int_en	UART interrupt enable
uart_status	Bus busy status
sts_urx_abr_prd	Aute baud rate detection
urx_abr_prd_b01	ABR detection bit 0/1
urx_abr_prd_b23	ABR detection bit 2/3
urx_abr_prd_b45	ABR detection bit 4/5
urx_abr_prd_b67	ABR detection bit 6/7
urx_abr_pw_tol	0x55 ABR max allowable error
urx_bcr_int_cfg	BCR interrupt counter
utx_rs485_cfg	RS-485 configure
uart_fifo_config_0	FIFO status and DMA mode
uart_fifo_config_1	FIFO threshold and available count
uart_fifo_wdata	TX FIFO
uart_fifo_rdata	RX FIFO

10.4.1 utx_config

Address: 0x40010000

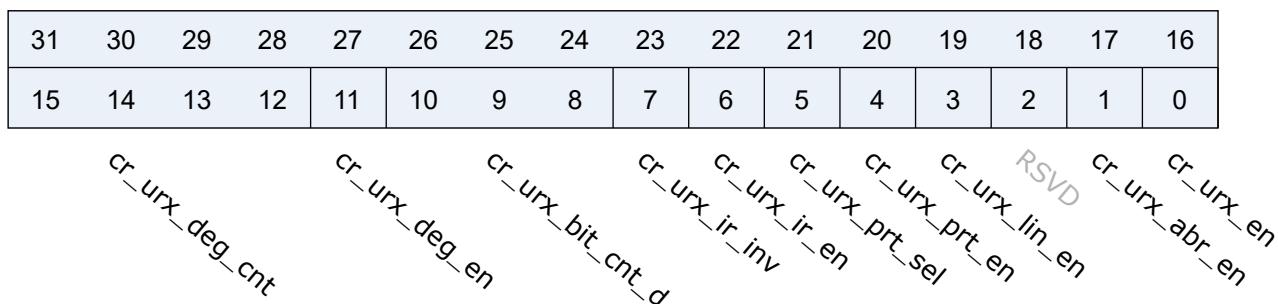


Bits	Name	Type	Reset	Description
31:16	cr_utx_len	r/w	16'd0	Length of UART TX data transfer (Unit: character/byte), TX end interrupt will be triggered when TX send cr_utx_len+1 bytes data (Don't-care if cr_utx_frm_en is enabled)
15:13	cr_utx_bit_cnt_b	r/w	3'd4	UART TX BREAK bit count (for LIN protocol) Note: Additional 8 bit times will be added since LIN Break field requires at least 13 bit times 4: 4+1+8=13 bit 5: 5+1+8=14 bit
12:11	cr_utx_bit_cnt_p	r/w	2'd1	UART TX STOP bit count (unit: 0.5 bit) 0: 0.5 bit 1: 1 bit 2: 1.5 bit 3: 2 bit
10:8	cr_utx_bit_cnt_d	r/w	3'd7	UART TX DATA bit count for each character 4: 5 bit 5: 6 bit 6: 7 bit 7: 8 bit
7	cr_utx_ir_inv	r/w	1'b0	Inverse signal of UART TX output in IR mode
6	cr_utx_ir_en	r/w	1'b0	Enable signal of UART TX IR mode
5	cr_utx_prt_sel	r/w	1'b0	Select signal of UART TX parity bit 1: Odd parity 0: Even parity
4	cr_utx_prt_en	r/w	1'b0	Enable signal of UART TX parity bit
3	cr_utx_lin_en	r/w	1'b0	Enable signal of UART TX LIN mode (LIN header will be sent before sending data)
2	cr_utx_frm_en	r/w	1'b0	Enable signal of UART TX freerun mode (utx_end_int will be disabled)
1	cr_utx_cts_en	r/w	1'b0	Enable signal of UART TX CTS flow control function
0	cr_utx_en	r/w	1'b0	Enable signal of UART TX function Asserting this bit will trigger the transaction, and should be de-asserted after finish

10.4.2 urx_config

Address: 0x40010004

cr_urx_len



Bits	Name	Type	Reset	Description
31:16	cr_urx_len	r/w	16'd0	Length of UART RX data transfer (Unit: character/byte) urx_end_int will assert when this length is reached
15:12	cr_urx_deg_cnt	r/w	4'd0	De-glitch function cycle count
11	cr_urx_deg_en	r/w	1'b0	Enable signal of RXD input de-glitch function
10:8	cr_urx_bit_cnt_d	r/w	3'd7	UART RX DATA bit count for each character, like cr_utx_bit_cnt_d
7	cr_urx_ir_inv	r/w	1'b0	Inverse signal of UART RX input in IR mode
6	cr_urx_ir_en	r/w	1'b0	Enable signal of UART RX IR mode
5	cr_urx_prt_sel	r/w	1'b0	Select signal of UART RX parity bit 1: Odd parity 0: Even parity
4	cr_urx_prt_en	r/w	1'b0	Enable signal of UART RX parity bit
3	cr_urx_lin_en	r/w	1'b0	Enable signal of UART RX LIN mode (LIN header will be required and checked before receiving data)
2	RSVD			
1	cr_urx_abr_en	r/w	1'b0	Enable signal of UART RX Auto Baud Rate detection function
0	cr_urx_en	r/w	1'b0	Enable signal of UART RX function

10.4.3 uart_bit_prd

Address: 0x40010008

cr_urx_bit_prd

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

cr_utx_bit_prd

Bits	Name	Type	Reset	Description
31:16	cr_urx_bit_prd	r/w	16'd255	Period of each UART RX bit, RX baud rate = uart clock / (cr_urx_bit_prd + 1)
15:0	cr_utx_bit_prd	r/w	16'd255	Period of each UART TX bit, TX baud rate = uart clock / (cr_utx_bit_prd + 1)

10.4.4 data_config

Address: 0x4001000c

RSVD																
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	

cr_uart_bit_inv

Bits	Name	Type	Reset	Description
31:1	RSVD			
0	cr_uart_bit_inv	r/w	1'b0	Bit-inverse signal for each data byte 0: Each byte is sent out LSB-first 1: Each byte is sent out MSB-first

10.4.5 utx_ir_position

Address: 0x40010010

cr_utx_ir_pos_p

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

cr_utx_ir_pos_s

Bits	Name	Type	Reset	Description
31:16	cr_utx_ir_pos_p	r/w	16'd159	STOP position of UART TX IR pulse
15:0	cr_utx_ir_pos_s	r/w	16'd112	START position of UART TX IR pulse

10.4.6 urx_ir_position

Address: 0x40010014

RSVD																
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	

cr_urx_ir_pos_s

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	cr_urx_ir_pos_s	r/w	16'd111	START position of UART RXD pulse recovered from IR signal

10.4.7 urx_rto_timer

Address: 0x40010018

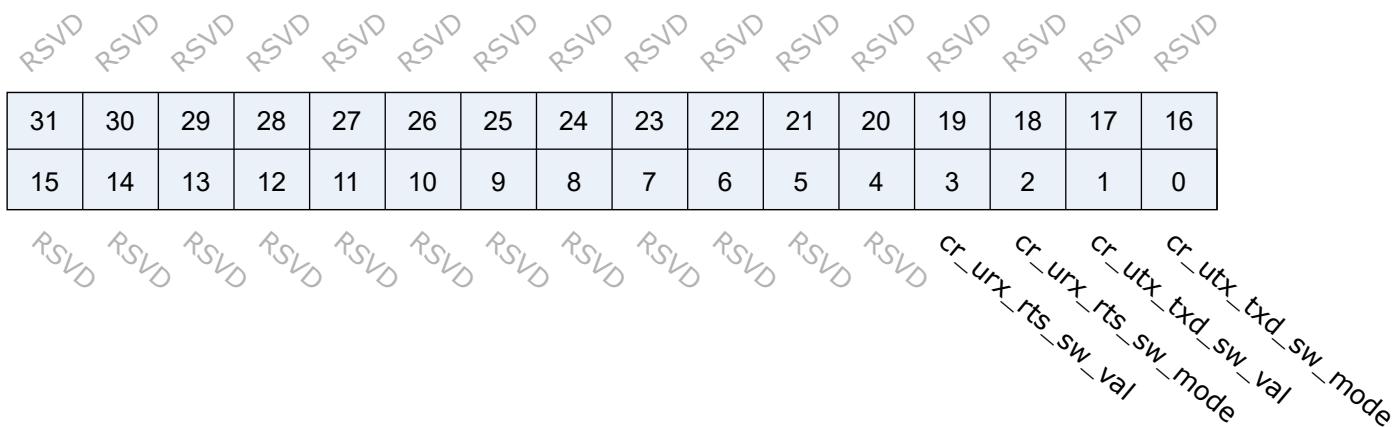
RSVD																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

cr_urx_rto_value

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	cr_urx_rto_value	r/w	8'd15	Time-out value for triggering RTO interrupt (unit: bit time)

10.4.8 uart_sw_mode

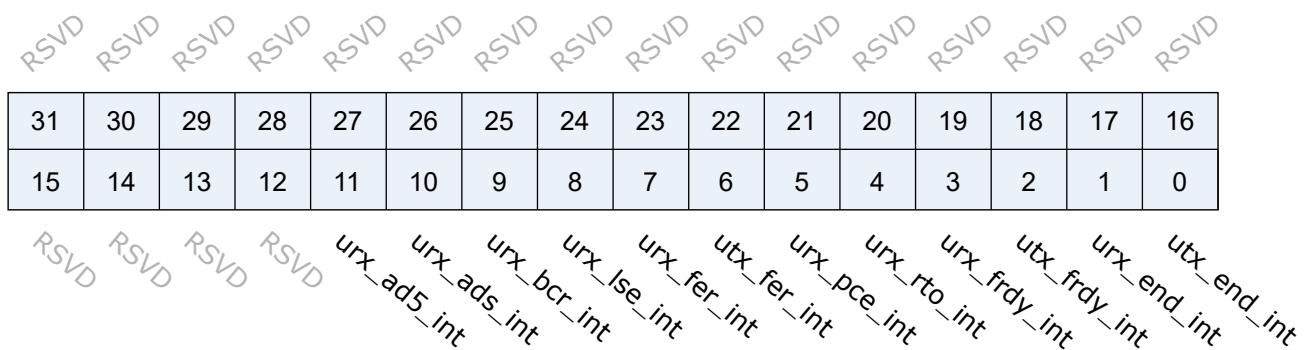
Address: 0x4001001c



Bits	Name	Type	Reset	Description
31:4	RSVD			
3	cr_urx_rts_sw_val	r/w	1'b0	UART RX RTS output SW control value 0: Low level 1: High level
2	cr_urx_rts_sw_mode	r/w	1'b0	UART RX RTS output SW control mode enable
1	cr_utx_txd_sw_val	r/w	1'b0	UART TX TXD output SW control value 0: Low level 1: High level
0	cr_utx_txd_sw_mode	r/w	1'b0	UART TX TXD output SW control mode enable

10.4.9 uart_int_sts

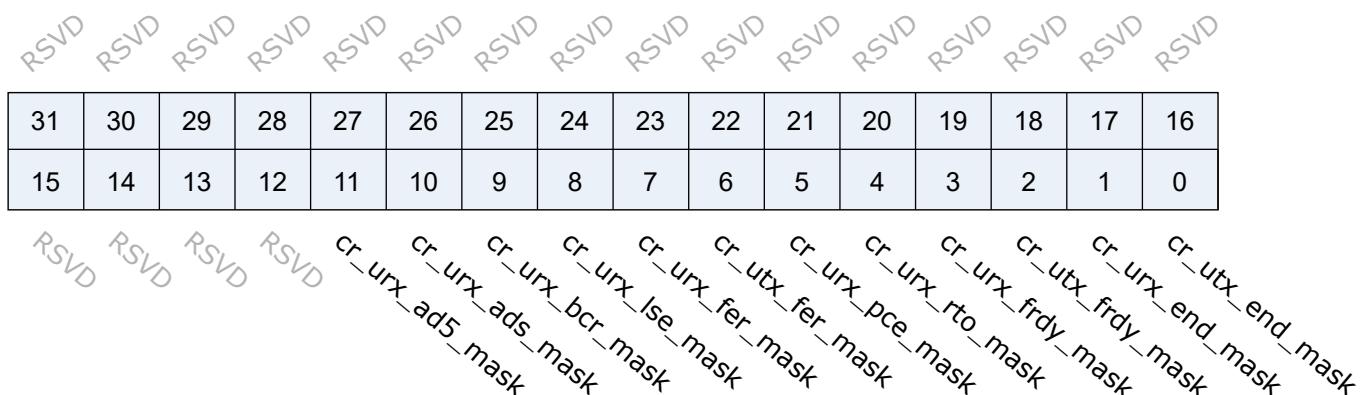
Address: 0x40010020



Bits	Name	Type	Reset	Description
31:12	RSVD			
11	urx_ad5_int	r	1'b0	UART RX ABR Detection finish interrupt using codeword 0x55
10	urx_ads_int	r	1'b0	UART RX ABR Detection finish interrupt using START bit
9	urx_bcr_int	r	1'b0	UART RX byte count reached interrupt
8	urx_lse_int	r	1'b0	UART RX LIN mode sync field error interrupt
7	urx_fer_int	r	1'b0	UART RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
6	utx_fer_int	r	1'b0	UART TX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
5	urx_pce_int	r	1'b0	UART RX parity check error interrupt
4	urx_rto_int	r	1'b0	UART RX Time-out interrupt
3	urx_frdy_int	r	1'b0	UART RX FIFO ready ($rx_fifo_cnt > rx_fifo_th$) interrupt, auto-cleared when data is popped
2	utx_frdy_int	r	1'b1	UART TX FIFO ready ($tx_fifo_cnt > tx_fifo_th$) interrupt, auto-cleared when data is pushed
1	urx_end_int	r	1'b0	UART RX transfer end interrupt (set according to cr_urx_len)
0	utx_end_int	r	1'b0	UART TX transfer end interrupt (set according to cr_utx_len)

10.4.10 uart_int_mask

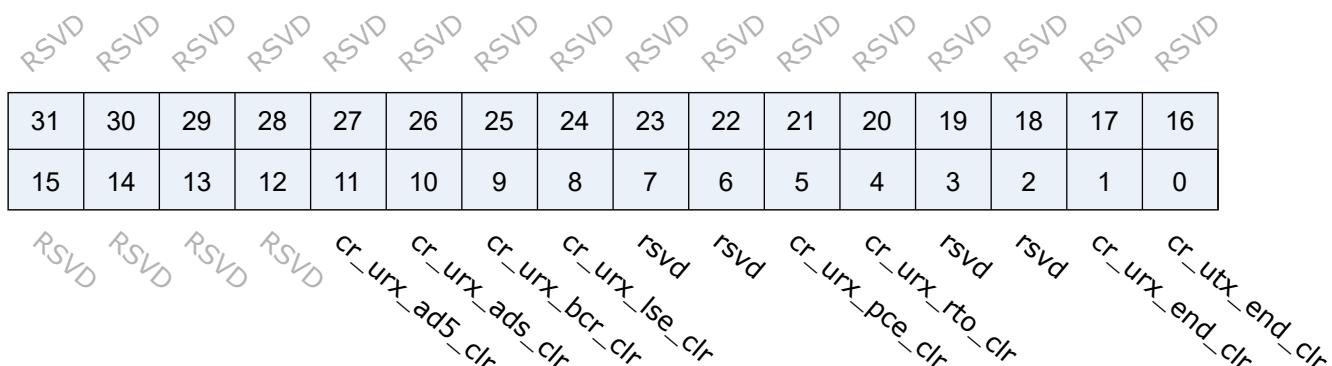
Address: 0x40010024



Bits	Name	Type	Reset	Description
31:12	RSVD			
11	cr_urx_ad5_mask	r/w	1'b1	Interrupt mask of urx_ad5_int
10	cr_urx_ads_mask	r/w	1'b1	Interrupt mask of urx_ads_int
9	cr_urx_bcr_mask	r/w	1'b1	Interrupt mask of urx_bcr_int
8	cr_urx_lse_mask	r/w	1'b1	Interrupt mask of urx_lse_int
7	cr_urx_fer_mask	r/w	1'b1	Interrupt mask of urx_fer_int
6	cr_utx_fer_mask	r/w	1'b1	Interrupt mask of utx_fer_int
5	cr_urx_pce_mask	r/w	1'b1	Interrupt mask of urx_pce_int
4	cr_urx_rto_mask	r/w	1'b1	Interrupt mask of urx_rto_int
3	cr_urx_frdy_mask	r/w	1'b1	Interrupt mask of urx_frdy_int
2	cr_utx_frdy_mask	r/w	1'b1	Interrupt mask of utx_frdy_int
1	cr_urx_end_mask	r/w	1'b1	Interrupt mask of urx_end_int
0	cr_utx_end_mask	r/w	1'b1	Interrupt mask of utx_end_int

10.4.11 uart_int_clear

Address: 0x40010028



Bits	Name	Type	Reset	Description
31:12	RSVD			
11	cr_urx_ad5_clr	w1c	1'b0	Interrupt clear of urx_ad5_int
10	cr_urx_ads_clr	w1c	1'b0	Interrupt clear of urx_ads_int
9	cr_urx_bcr_clr	w1c	1'b0	Interrupt clear of urx_bcr_int
8	cr_urx_lse_clr	w1c	1'b0	Interrupt clear of urx_lse_int
7	rsvd	rsvd	1'b0	
6	rsvd	rsvd	1'b0	
5	cr_urx_pce_clr	w1c	1'b0	Interrupt clear of urx_pce_int
4	cr_urx_rto_clr	w1c	1'b0	Interrupt clear of urx_rto_int
3	rsvd	rsvd	1'b0	
2	rsvd	rsvd	1'b0	
1	cr_urx_end_clr	w1c	1'b0	Interrupt clear of urx_end_int
0	cr_utx_end_clr	w1c	1'b0	Interrupt clear of utx_end_int

10.4.12 uart_int_en

Address: 0x4001002c

RSVD															
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

RSVD RSVD RSVD RSVD cr_urx_ad5_en cr_urx_ads_en cr_urx_bcr_en cr_urx_lse_en cr_utx_fer_en cr_utx_pce_en cr_urx_rto_en cr_urx_frdy_en cr_urx_end_en cr_utx_end_en

Bits	Name	Type	Reset	Description
31:12	RSVD			
11	cr_urx_ad5_en	r/w	1'b1	Interrupt enable of urx_ad5_int
10	cr_urx_ads_en	r/w	1'b1	Interrupt enable of urx_ads_int
9	cr_urx_bcr_en	r/w	1'b1	Interrupt enable of urx_bcr_int
8	cr_urx_lse_en	r/w	1'b1	Interrupt enable of urx_lse_int
7	cr_urx_fer_en	r/w	1'b1	Interrupt enable of urx_fer_int
6	cr_utx_fer_en	r/w	1'b1	Interrupt enable of utx_fer_int
5	cr_urx_pce_en	r/w	1'b1	Interrupt enable of urx_pce_int
4	cr_urx_rto_en	r/w	1'b1	Interrupt enable of urx_rto_int
3	cr_urx_frdy_en	r/w	1'b1	Interrupt enable of urx_frdy_int
2	cr_utx_frdy_en	r/w	1'b1	Interrupt enable of utx_frdy_int
1	cr_urx_end_en	r/w	1'b1	Interrupt enable of urx_end_int
0	cr_utx_end_en	r/w	1'b1	Interrupt enable of utx_end_int

10.4.13 uart_status

Address: 0x40010030

RSVD	sts_urx_bus_busy																								
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										

Bits	Name	Type	Reset	Description
31:2	RSVD			
1	sts_urx_bus_busy	r	1'b0	Indicator of UART RX bus busy 0: Idle 1: Busy
0	sts_utx_bus_busy	r	1'b0	Indicator of UART TX bus busy 0: Idle 1: Busy

10.4.14 sts_urx_abr_prd

Address: 0x40010034

sts_urx_abr_prd_0x55

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

sts_urx_abr_prd_start

Bits	Name	Type	Reset	Description
31:16	sts_urx_abr_prd_0x55	r	16'd0	Bit period of Auto Baud Rate detection using codeword 0x55, baud rate = uart clock / (sts_urx_abr_prd_0x55 + 1)
15:0	sts_urx_abr_prd_start	r	16'd0	Bit period of Auto Baud Rate detection using START bit, baud rate = uart clock / (sts_urx_abr_prd_start + 1)

10.4.15 urx_abr_prd_b01

Address: 0x40010038

sts_urx_abr_prd_bit1

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

sts_urx_abr_prd_bit0

Bits	Name	Type	Reset	Description
31:16	sts_urx_abr_prd_bit1	r	16'd0	Bit period of Auto Baud Rate detection - bit[1]
15:0	sts_urx_abr_prd_bit0	r	16'd0	Bit period of Auto Baud Rate detection - bit[0]

10.4.16 urx_abr_prd_b23

Address: 0x4001003c

sts_urx_abr_prd_bit3

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

sts_urx_abr_prd_bit2

Bits	Name	Type	Reset	Description
31:16	sts_urx_abr_prd_bit3	r	16'd0	Bit period of Auto Baud Rate detection - bit[3]
15:0	sts_urx_abr_prd_bit2	r	16'd0	Bit period of Auto Baud Rate detection - bit[2]

10.4.17 urx_abr_prd_b45

Address: 0x40010040

sts_urx_abr_prd_bit5

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

sts_urx_abr_prd_bit4

Bits	Name	Type	Reset	Description
31:16	sts_urx_abr_prd_bit5	r	16'd0	Bit period of Auto Baud Rate detection - bit[5]
15:0	sts_urx_abr_prd_bit4	r	16'd0	Bit period of Auto Baud Rate detection - bit[4]

10.4.18 urx_abr_prd_b67

Address: 0x40010044

sts_urx_abr_prd_bit7

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

sts_urx_abr_prd_bit6

Bits	Name	Type	Reset	Description
31:16	sts_urx_abr_prd_bit7	r	16'd0	Bit period of Auto Baud Rate detection - bit[7]
15:0	sts_urx_abr_prd_bit6	r	16'd0	Bit period of Auto Baud Rate detection - bit[6]

10.4.19 urx_abr_pw_tol

Address: 0x40010048

RSVD																
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	

cr_urx_abr_pw_tol

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	cr_urx_abr_pw_tol	r/w	8'd3	Auto Baud Rate detection pulse-width tolerance for using codeword 0x55

10.4.20 urx_bcr_int_cfg

Address: 0x40010050

sts_urx_bcr_count

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

cr_urx_bcr_value

Bits	Name	Type	Reset	Description
31:16	sts_urx_bcr_count	r	16'd0	Current byte count of urx_bcr_int counter, auto-cleared by cr_urx_bcr_clr
15:0	cr_urx_bcr_value	r/w	16'hFFFF	Byte count setting for urx_bcr_int counter

10.4.21 utx_rs485_cfg

Address: 0x40010054

RSVD																
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	

cr_utx_rs485_en
cr_utx_rs485_pol

Bits	Name	Type	Reset	Description
31:2	RSVD			
1	cr_utx_rs485_pol	r/w	1'b1	DE pin polarity in RS-485 transceiver mode 1'b0: DE is active-low 1'b1: DE is active-high
0	cr_utx_rs485_en	r/w	1'b0	Enable signal for RS-485 transceiver mode 1'b0: Disabled, normal UART 1'b1: Enabled, IO is connected to RS-485 transceiver and RTS_O becomes DE function

10.4.22 uart_fifo_config_0

Address: 0x40010080

RSVD															
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

RSVD RSVD

rx_fifo_underflow rx_fifo_overflow tx_fifo_underflow tx_fifo_overflow rx_fifo_clr tx_fifo_clr uart_dma_rx_en uart_dma_tx_en

Bits	Name	Type	Reset	Description
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO, RX FIFO will be empty when write 1 to this bit
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO, TX FIFO will be empty when write 1 to this bit
1	uart_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	uart_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

10.4.23 uart_fifo_config_1

Address: 0x40010084

RSVD	RSVD	RSVD	rx_fifo_th								RSVD	RSVD	RSVD	tx_fifo_th			
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		

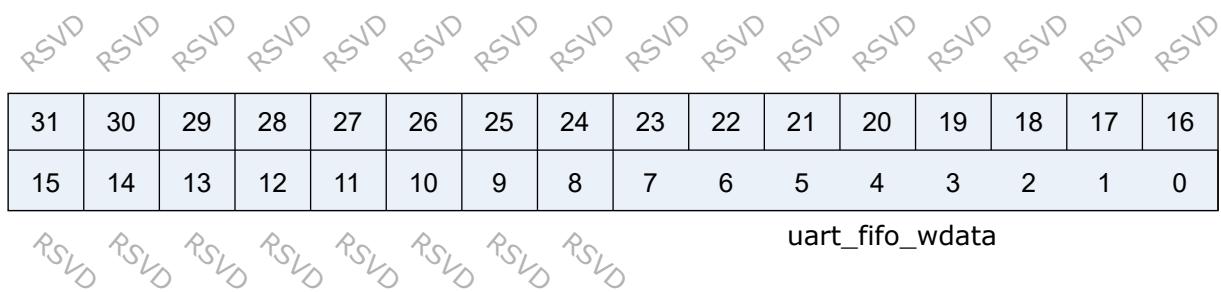
RSVD RSVD RSVD rx_fifo_th RSVD RSVD RSVD tx_fifo_th

RSVD RSVD rx_fifo_cnt RSVD RSVD tx_fifo_cnt

Bits	Name	Type	Reset	Description
31:29	RSVD			
28:24	rx_fifo_th	r/w	5'd0	RX FIFO threshold, dma_rx_req and rx_fifo_int will not be asserted if rx_fifo_cnt is less than this value
23:21	RSVD			
20:16	tx_fifo_th	r/w	5'd0	TX FIFO threshold, dma_tx_req and tx_fifo_int will not be asserted if tx_fifo_cnt is less than this value
15:14	RSVD			
13:8	rx_fifo_cnt	r	6'd0	RX FIFO available count, means byte count of data received in RX FIFO
7:6	RSVD			
5:0	tx_fifo_cnt	r	6'd32	TX FIFO available count, means empty space remained in TX FIFO

10.4.24 uart_fifo_wdata

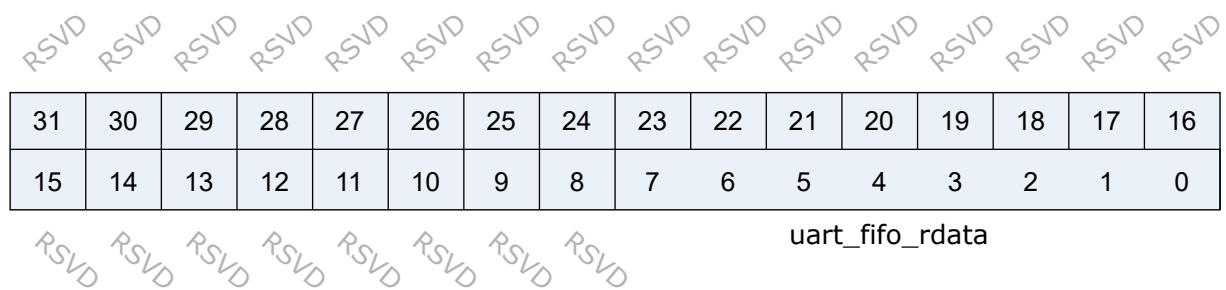
Address: 0x40010088



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	uart_fifo_wdata	w	x	TX FIFO, size is 32*1=32-byte

10.4.25 uart_fifo_rdata

Address: 0x4001008c



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	uart_fifo_rdata	r	8'h0	RX FIFO, size is 32*1=32-byte

11

I2C

11.1 Overview

Inter-Integrated Circuit (I2C) is a serial communication bus, which uses a multi-slave and multi-master architecture and is connected to low-speed peripherals. Each device has a unique address identifier and can be used as a transmitter or receiver. The address of each device connected to the bus can be set by software through a unique address and the existing master or slave relation. The master can work as a master transmitter or a master receiver. If two or more masters are initialized at the same time, data can be prevented from being damaged through conflict detection and arbitration during transmission.

BL616/BL618 has two I2C controller masters, whose slaveAddr, subAddr, and data to be transferred can be flexibly configured, to facilitate communication with slaves. With the FIFO of 2-word depth and interrupt function, it can be used with DMA to improve efficiency and supports flexible adjustment of clock frequency.

11.2 Features

- Master mode
- Multi-master mode and arbitration function
- Flexible adjustment of clock frequency
- Supports 10-bit address mode
- Supports DMA transfer mode
- Supports multiple interrupt mechanisms

11.3 Functional Description

Table 11.1: I2C pin list

Name	Type	Description
I2Cx_SCL	Input/output	I2C serial clock signal
I2Cx_SDA	Input/output	I2C serial data signal

11.3.1 Start and stop conditions

All transmissions start with a START condition and end with a STOP condition. The START and STOP conditions are generally generated by the master. The bus is considered to be busy after the START condition and to be idle for a certain period of time after the STOP condition.

START condition: SDA produces a high-to-low level transition when SCL is high;

STOP condition: SDA produces a low-to-high level transition when SCL is high.

Waveform diagram:

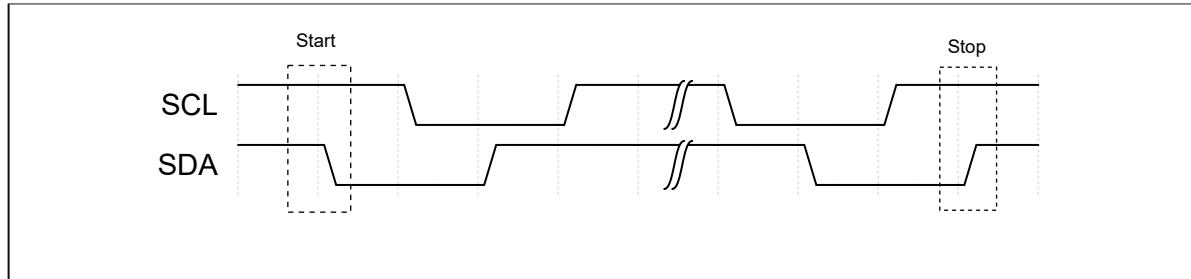


Fig. 11.1: Start and stop conditions of I2C

11.3.2 Data Transfer Format

7-bit address mode:

The first 8 bits transferred are addressing bytes, including a 7-bit slave address and a 1-bit direction bit. Sending or receiving data by the master is controlled by the 8th bit in the first byte sent by the master. If it is 0, it means that the data is sent by the master, while “1” indicates that data is received by the master, followed by the slave sending out an acknowledgement (ACK) bit. Upon data transfer completed, the master sends out a STOP signal, with waveform shown below:

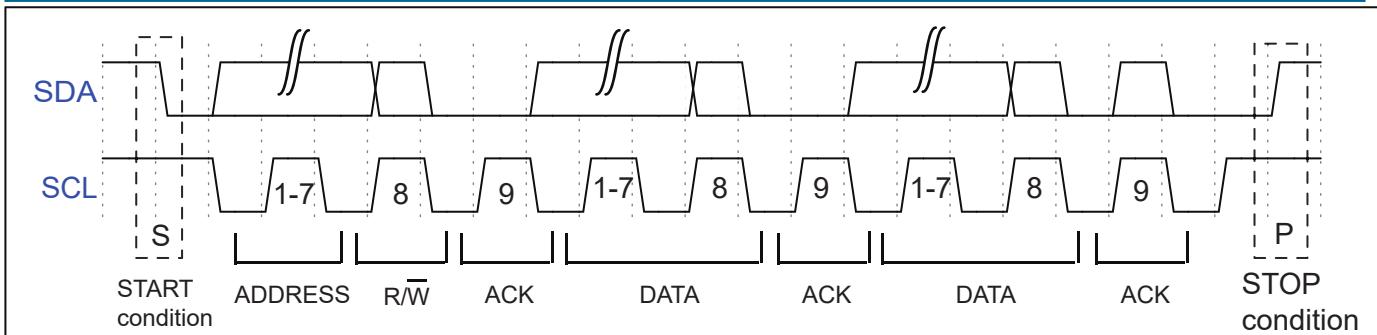


Fig. 11.2: I2C data transfer format

Master Transmit and Slave Receive Timing

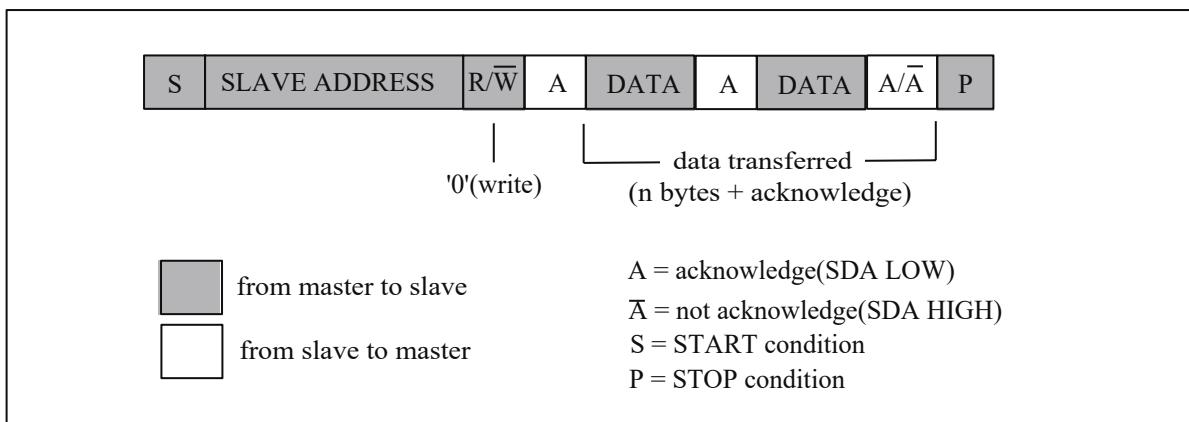


Fig. 11.3: Timing of master transmitter and slave receiver

Master Receive and Slave Transmit Timing

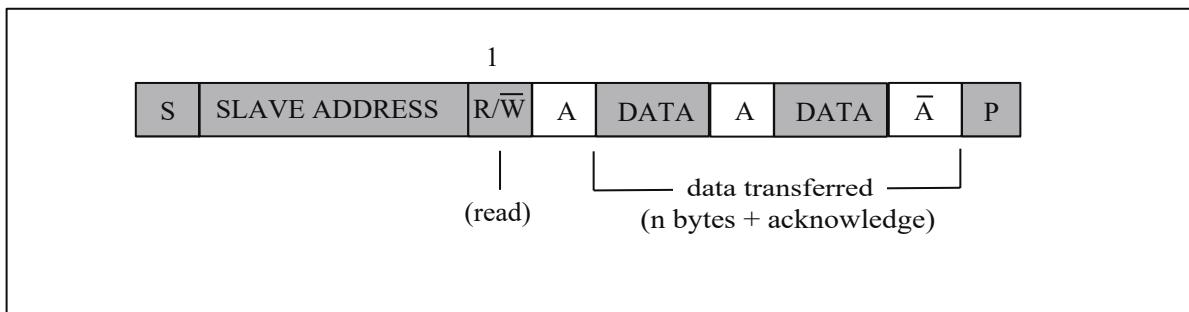


Fig. 11.4: Timing of master receiver and slave transmitter

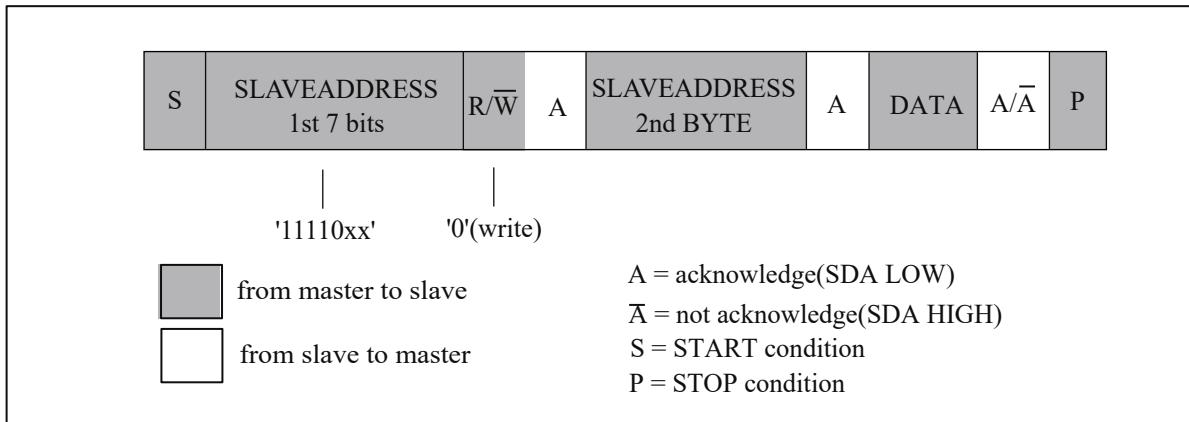
10-bit address mode: The cr_i2c_10b_addr_en in the register i2c_config must be set to 1 before use.

The 10-bit slave address consists of the two bytes after the START condition (S) or the repeated START condition (Sr). The first 7 bits of the first byte are 1111 0XX, where XX are the first two bits of MSB of the 10-bit address. The 8th bit of the first byte is the read/write bit that determines the transfer direction. Although there are 8 possible combinations

of 1111 XXX, only the four types of 1111 0XX can be used for 10-bit addressing, and the remaining four types of 1111 1XX are used for future I2C expansion.

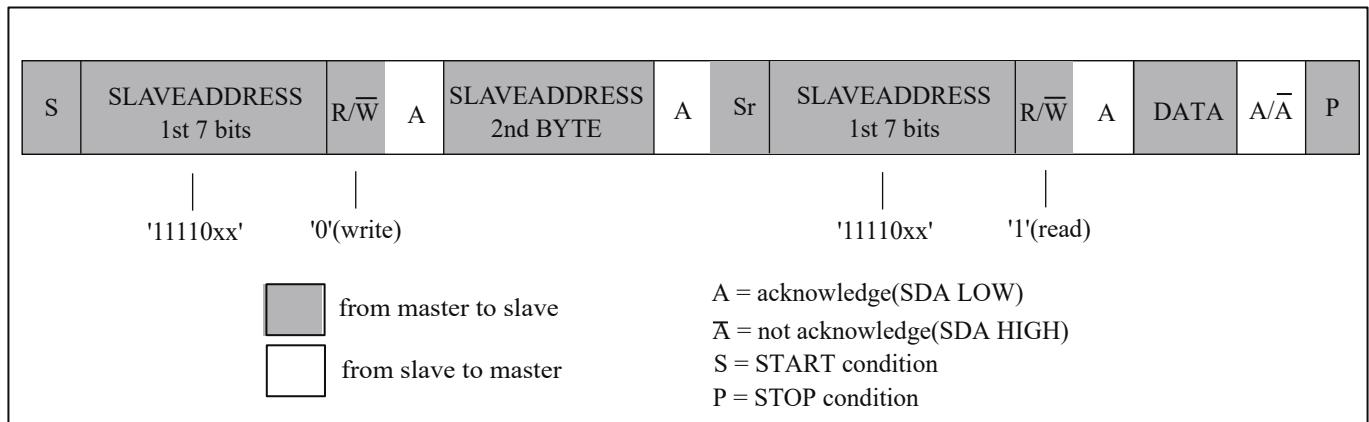
The aforementioned read-write formats for 7-bit addressing all suit 10-bit addressing, as follows:

1. A master-transmitter sends data to a slave-receiver with a 10-bit slave address



The figure shows that the transfer direction is unchanged. When receiving the 10-bit address following the START condition, the slave compares the first byte (1111 0XX) of the slave address with its own address, and checks whether the eighth bit (read/write bit) is 0. It is possible that multiple devices all match and generate an acknowledgement (A1). Next, all the slaves start to match their own addresses with the 8 bits of the second byte (XXXX XXXX). At this time, only one slave matches and generates an acknowledgement (A2). The slave that is addressed by the master will remain in the addressed state until it receives a STOP condition or a repeated START condition, followed by a different slave address.

2. A master-receiver receives data from a slave-transmitter (10-bit slave address)



The transfer direction will change after the second read/write bit. Before the second acknowledgement A2, the process is the same as that of the master-transmitter addressing the slave-receiver. After the repeated START condition (Sr), the matched slave will remain in the addressed state. This slave will check whether the first 7 bits of the first byte after Sr are correct, and then test whether the 8th bit is 1 (read). If this also matches, the slave considers that it is addressed as a transmitter and generates an acknowledgement (A3). The slave-transmitter will remain in the addressed state

until it receives the STOP condition (P) or the repeated START condition (Sr) followed by a different slave address. Then, under Sr, all the slaves will compare their addresses with 11110XX and test the eighth bit (read/write bit). However, they will not be addressed, because for 10-bit devices, the read/write bit is 1, or for 7-bit devices, the slave addresses of 1111 0XX do not match.

11.3.3 Arbitration

When there are multiple masters on I2C bus, it may happen that multiple masters start data transfer at the same time. At this time, the arbitration mechanism will decide which master has the right to transfer data, while other masters have to give up the control of the bus and wait until the bus is idle before transferring data again.

During data transfer, all masters must check whether the SDA is consistent with the data they want to send when SCL stays high. When the SDA level is different from the expected one, it means that other masters are transferring data at the same time. The masters with different SDA levels will lose the arbitration and other masters will complete the data transfer.

The waveform of two masters transferring data and initiating the arbitration mechanism at the same time is as follows:

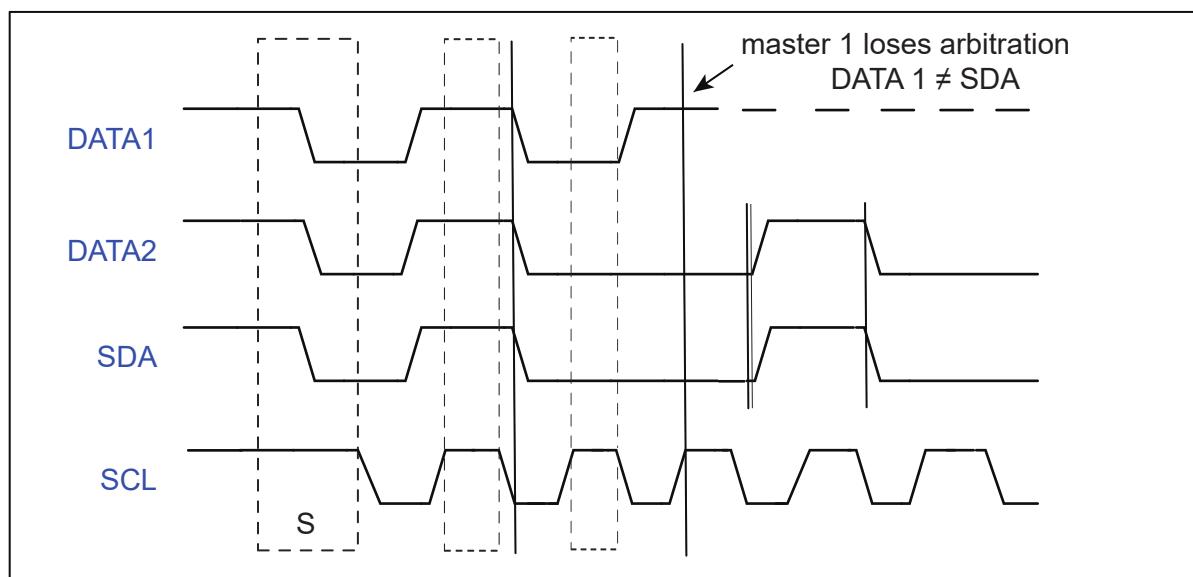


Fig. 11.5: Waveform of simultaneous data transfer

11.4 I2C Clock Setting

I2C clock can be derived from bclk (bus clock) and xclk, and frequency division can be done on this basis. The register i2c_prd_data can divide the clock of the data segment. The I2C module divides data transfer into 4 stages. Each stage is controlled by a single byte in the register, and the number of samples of each stage can be set. The 4 numbers jointly determine the division factor of i2c clock.

For example, bclk is 32M now, and the default value of the register i2c_prd_data is 0x0f0f0f0f without configuration, so I2C's clock frequency is $32M / ((15 + 1) * 4) = 500K$. Similarly, registers i2c_prd_start and i2c_prd_stop will divide the clock of the start and stop bits, respectively.

11.5 I2C Configuration Flow

11.5.1 Configuration Items

- Read/write flag bit
- Slave address
- Slave register address
- Slave register address length
- Data (TX: configure the sent data; RX: store the received data)
- Data length
- Enable signal

11.5.2 Read/Write Flag Bit

I2C supports TX and RX working statuses. The cr_i2c_pkt_dir in the register i2c_config represents the TX/RX status, “0” for TX status and “1” for RX status.

11.5.3 Slave Address

Each slave connected to I2C will have a unique device address, which is usually 7 bits long. This address will be written into the cr_i2c_slv_addr in the register i2c_config. I2C will automatically shift to the left by 1 bit before sending the address, and the TX/RX direction bit will be added to the LSB.

11.5.4 Slave Register Address

The slave register address represents the register address where I2C needs to read and write a slave register. The slave register address is written to the register i2c_sub_addr, and the cr_i2c_sub_addr_en in the register i2c_config must be set to 1. If cr_i2c_sub_addr_en in the register i2c_config is set to 0, the I2C master will skip the slave register address field when sending.

11.5.5 Slave Register Address Length

The slave register address length is subtracted by 1 and then written to cr_i2c_sub_addr_bc in the register i2c_config.

11.5.6 Data

It refers to the data that needs to be sent to or received from the slave. When sending data, I2C must write the data (in word) into the register i2c_fifo_wdata. When receiving data, I2C must read out the data (in word) from the register i2c_fifo_rdata.

11.5.7 Data Length

The data length is subtracted by 1 and then written to cr_i2c_pkt_len in the register i2c_config.

11.5.8 Enable Signal

After the above items are configured, when cr_i2c_m_en in the enable signal register i2c_config is set to 1, the I2C sending process will be started automatically.

When the read/write flag bit is configured as 0, I2C sends data and the master's transmission flow is as follows:

1. Start bit
2. (The slave address shifts to the left by 1 bit + 0) + ACK
3. Slave register address + ACK
4. 1-byte data + ACK
5. 1-byte data + ACK
6. Stop bit

When the read/write flag bit is configured as 1, I2C receives data and the master's transmission flow is as follows:

1. Start bit
2. (The slave address shifts to the left by 1 bit + 0) + ACK
3. Slave register address + ACK
4. Start bit
5. (The slave address shifts to the left by 1 bit + 1) + ACK
6. 1-byte data + ACK
7. 1-byte data + ACK
8. Stop bit

11.6 FIFO Management

I2C FIFO has a 2-word depth, and I2C includes RX FIFO and TX FIFO. The rx_fifo_cnt in the register i2c_fifo_config_1 represents how much data (in word) in RX FIFO needs to be read. The tx_fifo_cnt in the register i2c_fifo_config_1 represents how much free space (in word) in TX FIFO for writing.

I2C FIFO status:

- RX FIFO underflow: When the data in RX FIFO is completely read out or empty, if I2C continues to read data from RX FIFO, the rx_fifo_underflow in the register i2c_fifo_config_0 will be set to 1;
- RX FIFO overflow: When I2C receives data until the two words of RX FIFO are filled, without reading RX FIFO, if I2C receives data again, the rx_fifo_overflow in the register i2c_fifo_config_0 will be set to 1;
- TX FIFO underflow: When the data size filled into TX FIFO does not meet the configured I2C data length: cr_i2c_pkt_len in i2c_config, and no new data is filled into TX FIFO, the tx_fifo_underflow in the register i2c_fifo_config_0 will be set to 1;
- TX FIFO overflow: After the two words of TX FIFO are filled, before the data in TX FIFO is sent out, if data is filled into TX FIFO again, the tx_fifo_overflow in the register i2c_fifo_config_0 will be set to 1.

11.7 Use with DMA

I2C can send and receive data through DMA. Setting i2c_dma_tx_en in the register i2c_fifo_config_0 to 1 will enable the DMA TX mode. After the channel for I2C is allocated, DMA will transfer data from memory to the i2c_fifo_wdata register. Setting i2c_dma_rx_en in the register i2c_fifo_config_0 to 1 will enable the DMA RX mode. After the channel for I2C is allocated, DMA will transfer the data in the register i2c_fifo_rdata to memory. When I2C is used with DMA, DMA will automatically transfer data, so it is unnecessary for CPU to write data into I2C TX FIFO or read data from I2C RX FIFO.

11.7.1 DMA Sending Flow

1. Set read/write flag bit to 0
2. Set slave address
3. Set slave register address
4. Set slave register address length
5. Data Length
6. Set enable signal register to 1
7. Configure DMA transfer size
8. Configure the transfer width of DMA source address
9. Configure the transfer width of DMA destination address (when I2C is used with DMA, the transfer width of desti-

nation address must be set to 32 bits, which is word-aligned)

10. Configure the DMA source address as the memory address for storing sent data
11. Configure the DMA destination address to I2C TX FIFO address, i2c_fifo_wdata
12. Enable DMA

11.7.2 DMA Receiving Flow

1. Set read/write flag bit to 1
2. Set slave address
3. Set slave register address
4. Set slave register address length
5. Data Length
6. Set enable signal register to 1
7. Configure DMA transfer size
8. Configure the transfer width of DMA source address (when I2C is used with DMA, the transfer width of source address must be set to 32 bits, which is word-aligned)
9. Configure the transfer width of DMA destination address
10. Configure the DMA source address to I2C RX FIFO address, i2c_fifo_rdata
11. Configure the DMA destination address as the memory address for storing received data
12. Enable DMA

11.8 Interrupt

I2C includes the following interrupts:

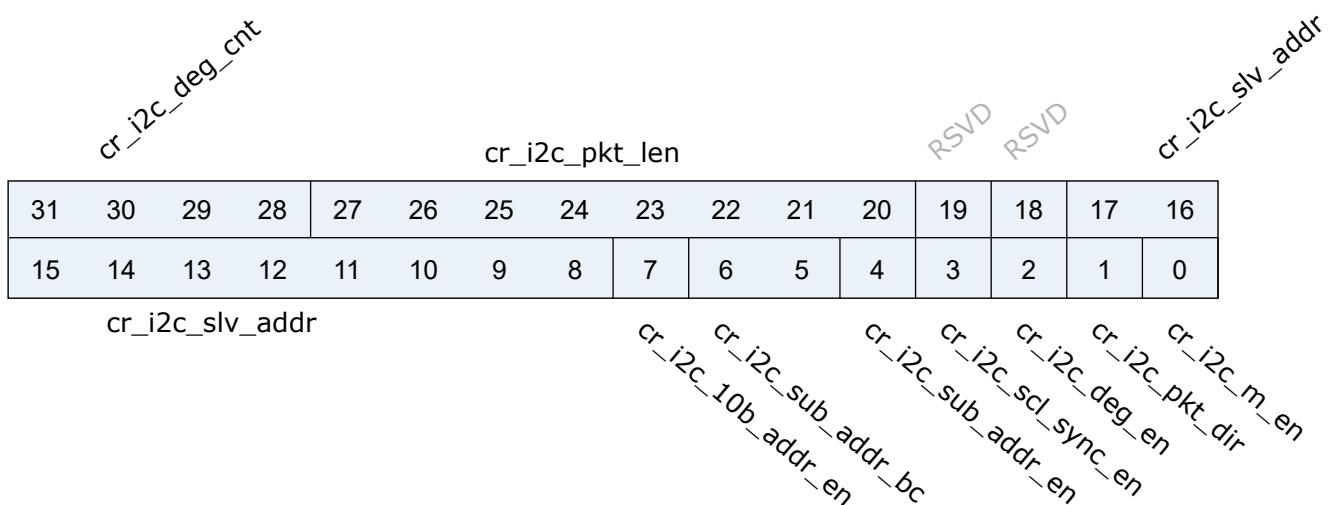
- I2C_TRANS_END_INT: I2C transfer end interrupt, which is generated when I2C completes a transfer
- I2C_TX_FIFO_READY_INT: When tx_fifo_cnt in i2c_fifo_config_1 is greater than tx_fifo_th, a TX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not satisfied
- I2C_RX_FIFO_READY_INT: When rx_fifo_cnt in i2c_fifo_config_1 is greater than rx_fifo_th, an RX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not satisfied
- I2C_NACK_RECV_INT: When the I2C module detects a NACK state, a NACK interrupt is generated
- I2C_ARB_LOST_INT: I2C arbitration lost interrupt
- I2C_FIFO_ERR_INT: FIFO ERROR interrupt is generated when TX/RX FIFO overflows or underflows

11.9 Register description

Name	Description
i2c_config	
i2c_int_sts	
i2c_sub_addr	
i2c_bus_busy	
i2c_prd_start	
i2c_prd_stop	
i2c_prd_data	
i2c_fifo_config_0	
i2c_fifo_config_1	
i2c_fifo_wdata	
i2c_fifo_rdata	

11.9.1 i2c_config

Address: 0x40018000



Bits	Name	Type	Reset	Description
31:28	cr_i2c_deg_cnt	r/w	4'd0	De-glitch function cycle count
27:20	cr_i2c_pkt_len	r/w	8'd0	Packet length (unit: byte)

Bits	Name	Type	Reset	Description
19:18	RSVD			
17:8	cr_i2c_slv_addr	r/w	10'h0	Slave address for I2C transaction (target address)
7	cr_i2c_10b_addr_en	r/w	1'b0	Slave address 10-bit mode enable
6:5	cr_i2c_sub_addr_bc	r/w	2'd0	Sub-address field byte count 2'd0: 1-byte, 2'd1: 2-byte, 2'd2: 3-byte, 2'd3: 4-byte
4	cr_i2c_sub_addr_en	r/w	1'b0	Enable signal of I2C sub-address field
3	cr_i2c_scl_sync_en	r/w	1'b1	Enable signal of I2C SCL synchronization, should be enabled to support Multi-Master and Clock-Stretching (Normally should not be turned-off)
2	cr_i2c_deg_en	r/w	1'b0	Enable signal of I2C input de-glitch function (for all input pins)
1	cr_i2c_pkt_dir	r/w	1'b1	Transfer direction of the packet 1'b0: Write; 1'b1: Read
0	cr_i2c_m_en	r/w	1'b0	Enable signal of I2C Master function Asserting this bit will trigger the transaction, and should be de-asserted after finish

11.9.2 i2c_int_sts

Address: 0x40018004

RSVD	RSVD	cr_i2c_fer_en	cr_i2c_arb_en	cr_i2c_nak_en	cr_i2c_rxf_en	cr_i2c_txf_en	cr_i2c_end_en	RSVD	RSVD	rsvd	cr_i2c_arb_clr	cr_i2c_nak_clr	rsvd	rsvd	cr_i2c_end_clr
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

RSVD RSVD cr_i2c_fer_mask cr_i2c_arb_mask cr_i2c_nak_mask cr_i2c_rxf_mask cr_i2c_txf_end_mask RSVD RSVD i2c_fer_int i2c_arb_int i2c_nak_int i2c_rxf_int i2c_txf_int i2c_end_int

Bits	Name	Type	Reset	Description
31:30	RSVD			
29	cr_i2c_fer_en	r/w	1'b1	Interrupt enable of i2c_fer_int
28	cr_i2c_arb_en	r/w	1'b1	Interrupt enable of i2c_arb_int

Bits	Name	Type	Reset	Description
27	cr_i2c_nak_en	r/w	1'b1	Interrupt enable of i2c_nak_int
26	cr_i2c_rxf_en	r/w	1'b1	Interrupt enable of i2c_rxf_int
25	cr_i2c_txf_en	r/w	1'b1	Interrupt enable of i2c_txf_int
24	cr_i2c_end_en	r/w	1'b1	Interrupt enable of i2c_end_int
23:22	RSVD			
21	rsvd	rsvd	1'b0	
20	cr_i2c_arb_clr	w1c	1'b0	Interrupt clear of i2c_arb_int
19	cr_i2c_nak_clr	w1c	1'b0	Interrupt clear of i2c_nak_int
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_i2c_end_clr	w1c	1'b0	Interrupt clear of i2c_end_int
15:14	RSVD			
13	cr_i2c_fer_mask	r/w	1'b1	Interrupt mask of i2c_fer_int
12	cr_i2c_arb_mask	r/w	1'b1	Interrupt mask of i2c_arb_int
11	cr_i2c_nak_mask	r/w	1'b1	Interrupt mask of i2c_nak_int
10	cr_i2c_rxf_mask	r/w	1'b1	Interrupt mask of i2c_rxf_int
9	cr_i2c_txf_mask	r/w	1'b1	Interrupt mask of i2c_txf_int
8	cr_i2c_end_mask	r/w	1'b1	Interrupt mask of i2c_end_int
7:6	RSVD			
5	i2c_fer_int	r	1'b0	I2C TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
4	i2c_arb_int	r	1'b0	I2C arbitration lost interrupt
3	i2c_nak_int	r	1'b0	I2C NACK-received interrupt
2	i2c_rxf_int	r	1'b0	I2C RX FIFO ready ($rx_fifo_cnt > rx_fifo_th$) interrupt, auto-cleared when data is popped
1	i2c_txf_int	r	1'b1	I2C TX FIFO ready ($tx_fifo_cnt > tx_fifo_th$) interrupt, auto-cleared when data is pushed
0	i2c_end_int	r	1'b0	I2C transfer end interrupt

11.9.3 i2c_sub_addr

Address: 0x40018008

cr_i2c_sub_addr_b3								cr_i2c_sub_addr_b2							
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
cr_i2c_sub_addr_b1								cr_i2c_sub_addr_b0							

Bits	Name	Type	Reset	Description
31:24	cr_i2c_sub_addr_b3	r/w	8'd0	I2C sub-address field - byte[3]
23:16	cr_i2c_sub_addr_b2	r/w	8'd0	I2C sub-address field - byte[2]
15:8	cr_i2c_sub_addr_b1	r/w	8'd0	I2C sub-address field - byte[1]
7:0	cr_i2c_sub_addr_b0	r/w	8'd0	I2C sub-address field - byte[0] (sub-address starts from this byte)

11.9.4 i2c_bus_busy

Address: 0x4001800c

RSVD																
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	
RSVD	sts_i2c_bus_busy_clr															

Bits	Name	Type	Reset	Description
31:2	RSVD			
1	cr_i2c_bus_busy_clr	w1c	1'b0	Clear signal of bus_busy status, not for normal usage (in case I2C bus hangs)
0	sts_i2c_bus_busy	r	1'b0	Indicator of I2C bus busy

11.9.5 i2c_prd_start

Address: 0x40018010

cr_i2c_prd_s_ph_3

cr_i2c_prd_s_ph_2

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

cr_i2c_prd_s_ph_1

cr_i2c_prd_s_ph_0

Bits	Name	Type	Reset	Description
31:24	cr_i2c_prd_s_ph_3	r/w	8'd15	Length of START condition phase 3
23:16	cr_i2c_prd_s_ph_2	r/w	8'd15	Length of START condition phase 2
15:8	cr_i2c_prd_s_ph_1	r/w	8'd15	Length of START condition phase 1
7:0	cr_i2c_prd_s_ph_0	r/w	8'd15	Length of START condition phase 0

11.9.6 i2c_prd_stop

Address: 0x40018014

cr_i2c_prd_p_ph_3

cr_i2c_prd_p_ph_2

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

cr_i2c_prd_p_ph_1

cr_i2c_prd_p_ph_0

Bits	Name	Type	Reset	Description
31:24	cr_i2c_prd_p_ph_3	r/w	8'd15	Length of STOP condition phase 3
23:16	cr_i2c_prd_p_ph_2	r/w	8'd15	Length of STOP condition phase 2
15:8	cr_i2c_prd_p_ph_1	r/w	8'd15	Length of STOP condition phase 1
7:0	cr_i2c_prd_p_ph_0	r/w	8'd15	Length of STOP condition phase 0

11.9.7 i2c_prd_data

Address: 0x40018018

cr_i2c_prd_d_ph_3

cr_i2c_prd_d_ph_2

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

cr_i2c_prd_d_ph_1

cr_i2c_prd_d_ph_0

Bits	Name	Type	Reset	Description
31:24	cr_i2c_prd_d_ph_3	r/w	8'd15	Length of DATA phase 3
23:16	cr_i2c_prd_d_ph_2	r/w	8'd15	Length of DATA phase 2
15:8	cr_i2c_prd_d_ph_1	r/w	8'd15	Length of DATA phase 1 Note: This value should not be set to 8'd0, adjust source clock rate instead if higher I2C clock rate is required
7:0	cr_i2c_prd_d_ph_0	r/w	8'd15	Length of DATA phase 0

11.9.8 i2c_fifo_config_0

Address: 0x40018080

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

RSVD RSVD

rx_fifo_underflow *rx_fifo_overflow* *tx_fifo_underflow* *tx_fifo_overflow* *tx_fifo_clr* *i2c_dma_rx_en* *i2c_dma_tx_en*

Bits	Name	Type	Reset	Description
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	i2c_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	i2c_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

11.9.9 i2c_fifo_config_1

Address: 0x40018084

RSVD	tx_fifo_th	RSVD	tx_fifo_th												
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

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	rx_fifo_cnt	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	tx_fifo_cnt
------	------	------	------	------	------	-------------	------	------	------	------	------	------	-------------

Bits	Name	Type	Reset	Description
31:25	RSVD			
24	rx_fifo_th	r/w	1'd0	RX FIFO threshold, dma_rx_req will not be asserted if rx_fifo_cnt is less than this value
23:17	RSVD			
16	tx_fifo_th	r/w	1'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:10	RSVD			
9:8	rx_fifo_cnt	r	2'd0	RX FIFO available count
7:2	RSVD			
1:0	tx_fifo_cnt	r	2'd2	TX FIFO available count

11.9.10 i2c_fifo_wdata

Address: 0x40018088

i2c_fifo_wdata

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

i2c_fifo_wdata

Bits	Name	Type	Reset	Description
31:0	i2c_fifo_wdata	w	x	

11.9.11 i2c_fifo_rdata

Address: 0x4001808c

i2c_fifo_rdata

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

i2c_fifo_rdata

Bits	Name	Type	Reset	Description
31:0	i2c_fifo_rdata	r	32'h0	

12

PWM

12.1 Overview

Pulse Width Modulation (PWM), an efficient technique that uses the digital signal of microprocessor to control the analog circuit, is widely used in fields like measurement, communication, and power control and conversion.

12.2 Features

- Two PWM signals are generated, each containing 4-channel PWM signal outputs, with 2 pairs of complementary PWM per channel
- There are optional three clock sources (bus clock<bclk>, crystal oscillator clock<xtal_ck>, and slow clock<32k>), with 16-bit clock divider
- Each PWM can be independently set to different cycles
- Each PWM channel has double threshold setting, allowing different duty ratios and phases, with more flexible pulse
- Each PWM channel has independent dead time setting
- Different active levels can be set for each PWM output pin
- Each PWM has an independent linked switch to connect/disconnect with the internal counter, and you can set the default output level when disconnected
- The software and external brake signals can set the PWM output level to a preset state
- Up to 9 trigger sources for triggering ADC conversion
- Interrupts will be generated when these events occur: counter overflow, threshold comparison and matching, brake signal generation, and PWM cycles reaching the set value.

12.3 Functional Description

12.3.1 Clock and Frequency Divider

Each PWM counter has three optional clock sources:

1. bclk—bus clock of chip
2. XTAL—external crystal oscillator clock
3. f32k—RTC clock of system

Each counter has its own 16-bit frequency divider, which can divide the selected clock through APB. The PWM counter will take the divided clock as the counting cycle unit, and add one every time a counting cycle ends.

12.3.2 Active Level

Different application scenarios require different active level states, some being active at low level and others being active at high level. The active level of each PWM output can be set independently.

When the `<pwm_chy_ppl>` bit in the register `pwm_mcx_config1` is set to 1, it means that the forward channel of channel y ($y = 0, 1, 2, 3$) of `pwmx` ($x = 0$) is set to be active at high level. That is, when the PWM module outputs logical ‘1’, its corresponding pin outputs high level, and when it outputs logical ‘0’, that outputs low level. When `<pwm_chy_ppl>` is set to 0, it means that the same is set to be active at low level. That is, when the PWM module outputs logical ‘1’, its corresponding pin outputs low level, and when it outputs logical ‘0’, that outputs high level. The setting bit of the complementary channel is `<pwm_chy_npl>`, and its setting rule is the same as that of the forward channel.

12.3.3 Principle of Pulse Generation

When the `<pwm_chy_pen>` bit in the register `pwm_mcx_config1` is set to 0, it means that the forward channel of channel y ($y = 0, 1, 2, 3$) of `pwmx` ($x = 0$) is set to a determined logic state, and then the state is determined by the `<pwm_chy_psi>` bit. When `<pwm_chy_psi>` is 0, the corresponding forward channel pin outputs logical ‘0’, and when the same is 1, that outputs logical ‘1’. The actual output level must be jointly determined with `<pwm_chy_ppl>`. The setting bit of the complementary channel is `<pwm_chy_nen>`, and its setting rule is the same as that of the forward channel.

When the `<pwm_chy_pen>` (`<pwm_chy_nen>`) bit in the register `pwm_mcx_config1` is set to 1, the forward (complementary) channel output of channel y ($y = 0, 1, 2, 3$) of `pwmx` ($x = 0$) is determined by comparing the internal counter with the two thresholds. If the counter value is between the two thresholds, the forward channel outputs logical ‘1’, while the complementary one outputs logical ‘0’. If the counter value is beyond the two thresholds, the forward channel outputs logical ‘0’, while the complementary one outputs logical ‘1’. The waveform is shown as follows.

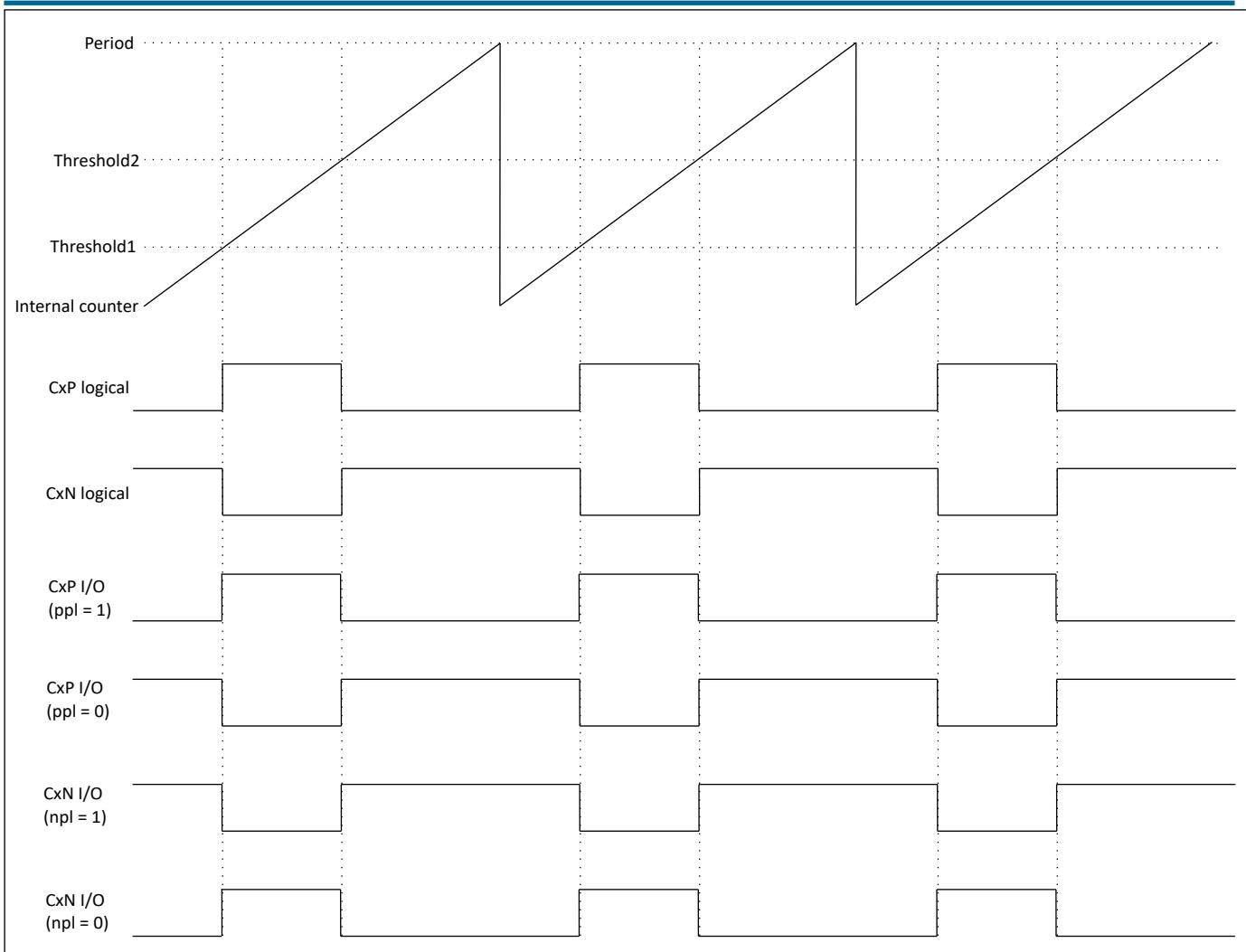


Fig. 12.1: Waveform of PWM in different configurations

12.3.4 Brake

When the bit <pwm_sw_break_en> of pwm_mcx_config0 (x=0) is 1, the brake signal is generated, and the level of the PWM output will be modified to the preset state. The preset state is set by the bits <pwm_chy_pbs> and <pwm_chy_nbs> (y=0,1,2,3) in the pwm_mcx_config1 (x=0) register. When this bit is set to 1, after the brake signal is generated, the PWM output logic 1, when this bit is set to 0, this PWM output logic 0 after the brake signal is generated. When <pwm_sw_break_en> is 0, it exits the braking state, and the PWM resumes the previous operating mode. The brake signal does not trigger an interrupt.

12.3.5 Dead Zone

Different dead time can be set for each PWM channel independently. When the value of dead time is not 0, the forward channel of PWM will delay the generation of jump level when it matches the threshold 1, and immediately change the level state when it matches the threshold 2. The complementary channel of PWM will change the level state immediately when it matches the threshold 1, and delay the generation of jump level when it matches the threshold 2. The length of dead zone is set by the register `pwm_mcx_dead_time`.

12.3.6 Cycle and Duty Ratio Calculation

The PWM cycle is determined the clock division factor and the clock duration cycle. The clock division factor is set by the register `PWMx_CLK_DIV[15:0](x: 0)` and is used to divide the source clock of PWM. The clock duration cycle is set by the register `PWMx_PERIOD[15:0](x: 0~1)` and is used to set how many divided clock cycles a PWM cycle consists of. That is, $\text{PWM cycle} = \text{PWM source clock}/\text{PWMx_CLK_DIV}[15:0]/\text{PWMx_PERIOD}[15:0]$.

12.3.7 PWM Interrupt

For each PWM, the cycle count value can be set by the high 16 bits of the register `pwm_mcx_period`. When the number of PWM cycles reaches this value, the PWM interrupt will be generated.

When the PWM count value reaches the number of cycles or it matches the threshold, the PWM interrupt will be generated.

12.3.8 ADC Linkage

When the counter matches the threshold or the count value reaches the number of cycles, it will generate the signal that internally triggers ADC startup conversion. It should be noted that only PWM0 can trigger such conversion, while PWM1 cannot do that. The specific trigger source is set by the `<pwm_adc_trg_src>` bit in the register `pwm_mcx_config0`. This feature is commonly used in timing sampling. The following is an example.

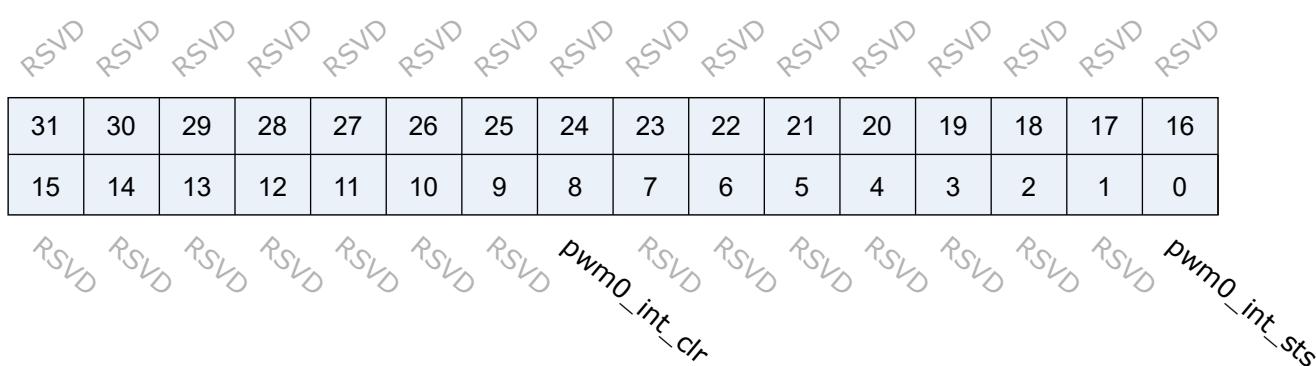
Application scenario: In BLDC application, there is such a requirement that the current flowing through the coil shall be detected while the motor speed is controlled by PWM. In a PWM cycle, after PWM controls the power device to turn on, the current becomes stable after a certain period of time. Then, it is necessary to sample the current value, which means that there is a strict phase difference between the time point of triggering ADC conversion and PWM. For example, if the channel 0 of PWM is used to drive one of the phases of the motor, and a 10 KHz square wave with a duty ratio of 20% needs to be generated, and ADC sampling needs to be performed at the middle time point of the high level of the square wave, then the PWM cycle is 100us. When the clock source is 1 MHz, the cycle count value is 100, and the two thresholds of channel 1 can be set to 0 and 20 respectively. At this time, the counter is between 0 and 20, and PWM outputs a high level, and otherwise it outputs a low level. When the threshold L of channel 2 is set to 10 and the `pwm_adc_trg_src` in the register `pwm_mc0_config0` is set to 4, `pwm_ch2l_int` can trigger ADC conversion. When the counter counts to 10, ADC will start sampling conversion at the middle time point of the high level generated by channel 1. This ensures that each sampling can meet the exact time requirement without CPU intervention, thus improving the performance.

12.4 Register description

Name	Description
pwm_int_config	
pwm_mc0_config0	
pwm_mc0_config1	
pwm_mc0_period	
pwm_mc0_dead_time	
pwm_mc0_ch0_thre	
pwm_mc0_ch1_thre	
pwm_mc0_ch2_thre	
pwm_mc0_ch3_thre	
pwm_mc0_int_sts	
pwm_mc0_int_mask	
pwm_mc0_int_clear	
pwm_mc0_int_en	

12.4.1 pwm_int_config

Address: 0x4000a400



Bits	Name	Type	Reset	Description
31:9	RSVD			
8	pwm0_int_clr	w1c	1'b0	PWM 0 interrupt clear (clear pwm_mc0_int_sts[10:0] all)

Bits	Name	Type	Reset	Description
7:1	RSVD			
0	pwm0_int_sts	r	1'b0	PWM 0 interrupt status (Check pwm_mc0_int_sts for detailed interrupt status)

12.4.2 pwm_mc0_config0

Address: 0x4000a440

reg_clk_sel	pwm_sts_stop	pwm_stop_mode	pwm_stop_en	pwm_ext_break_pl	pwm_ext_break_en	pwm_sw_break_en	pwm_adc_trg_src	pwm_stop_on_rept	RSVD	RSVD	RSVD
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

pwm_clk_div

Bits	Name	Type	Reset	Description
31:30	reg_clk_sel	r/w	2'd0	PWM clock source select, 2'b00-xclk ; 2'b01-bclk ; others-f32k_clk
29	pwm_sts_stop	r	1'b0	PWM stop status
28	pwm_stop_mode	r/w	1'b1	PWM stop mode, 1'b1 - graceful ; 1'b0 - abrupt
27	pwm_stop_en	r/w	1'b0	PWM stop enable
26	pwm_ext_break_pl	r/w	1'b0	PWM external break source polarity 1'b0: Active-LOW 1'b1: Active-HIGH
25	pwm_ext_break_en	r/w	1'b0	PWM external break source enable 1'b0: Disabled, external break signal is masked 1'b1: Enabled, external break signal is effective
24	pwm_sw_break_en	r/w	1'b0	PWM break enable 1'b0: Disabled, normal operation 1'b1: Enabled, PWM output will be determined by CxPBS/CxNBS

Bits	Name	Type	Reset	Description
23:20	pwm_adc_trg_src	r/w	4'hF	Select signal of ADC triggering source 4'd0: pwm_ch0l_int (Channel 0 ThresholdL reached) 4'd1: pwm_ch0h_int (Channel 0 ThresholdH reached) 4'd2: pwm_ch1l_int (Channel 1 ThresholdL reached) 4'd3: pwm_ch1h_int (Channel 1 ThresholdH reached) 4'd4: pwm_ch2l_int (Channel 2 ThresholdL reached) 4'd5: pwm_ch2h_int (Channel 2 ThresholdH reached) 4'd6: pwm_ch3l_int (Channel 3 ThresholdL reached) 4'd7: pwm_ch3h_int (Channel 3 ThresholdH reached) 4'd8: pwm_prde_int (Period End reached) Others: Disabled
19	pwm_stop_on_rept	r/w	1'b0	PWM stopped when rept_int is asserted 1'b0: Disabled, PWM keeps running when rept_int is asserted 1'b1: Enabled, PWM stops when rept_int is asserted. Clear rept_int to restore operation
18:16	RSVD			
15:0	pwm_clk_div	r/w	16'b0	PWM clock division

12.4.3 pwm_mc0_config1

Address: 0x4000a444

pwm_ch3_nbs	pwm_ch3_pbs	pwm_ch2_nbs	pwm_ch2_pbs	pwm_ch1_nbs	pwm_ch1_pbs	pwm_ch0_nbs	pwm_ch0_pbs	pwm_ch3_npl	pwm_ch3_ppl	pwm_ch2_npl	pwm_ch2_ppl	pwm_ch1_npl	pwm_ch1_ppl	pwm_ch0_npl	pwm_ch0_ppl
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
pwm_ch3_nsi	pwm_ch3_nen	pwm_ch3_psi	pwm_ch3_pen	pwm_ch2_nsi	pwm_ch2_nen	pwm_ch2_psi	pwm_ch2_pen	pwm_ch1_nsi	pwm_ch1_nen	pwm_ch1_psi	pwm_ch1_pen	pwm_ch0_nsi	pwm_ch0_nen	pwm_ch0_psi	pwm_ch0_pen

Bits	Name	Type	Reset	Description
31	pwm_ch3_nbs	r/w	1'b0	PWM channel 3 negative break state
30	pwm_ch3_pbs	r/w	1'b0	PWM channel 3 positive break state
29	pwm_ch2_nbs	r/w	1'b0	PWM channel 2 negative break state

Bits	Name	Type	Reset	Description
28	pwm_ch2_pbs	r/w	1'b0	PWM channel 2 positive break state
27	pwm_ch1_nbs	r/w	1'b0	PWM channel 1 negative break state
26	pwm_ch1_pbs	r/w	1'b0	PWM channel 1 positive break state
25	pwm_ch0_nbs	r/w	1'b0	PWM channel 0 negative break state
24	pwm_ch0_pbs	r/w	1'b0	PWM channel 0 positive break state
23	pwm_ch3_npl	r/w	1'b1	PWM channel 3 negative polarity
22	pwm_ch3_ppl	r/w	1'b1	PWM channel 3 positive polarity
21	pwm_ch2_npl	r/w	1'b1	PWM channel 2 negative polarity
20	pwm_ch2_ppl	r/w	1'b1	PWM channel 2 positive polarity
19	pwm_ch1_npl	r/w	1'b1	PWM channel 1 negative polarity
18	pwm_ch1_ppl	r/w	1'b1	PWM channel 1 positive polarity
17	pwm_ch0_npl	r/w	1'b1	PWM channel 0 negative polarity
16	pwm_ch0_ppl	r/w	1'b1	PWM channel 0 positive polarity
15	pwm_ch3_nsi	r/w	1'b1	PWM channel 3 negative set idle state
14	pwm_ch3_nen	r/w	1'b0	PWM channel 3 negative enable pwm out
13	pwm_ch3_psi	r/w	1'b0	PWM channel 3 positive set idle state
12	pwm_ch3_pen	r/w	1'b0	PWM channel 3 positive enable pwm out
11	pwm_ch2_nsi	r/w	1'b1	PWM channel 2 negative set idle state
10	pwm_ch2_nen	r/w	1'b0	PWM channel 2 negative enable pwm out
9	pwm_ch2_psi	r/w	1'b0	PWM channel 2 positive set idle state
8	pwm_ch2_pen	r/w	1'b0	PWM channel 2 positive enable pwm out
7	pwm_ch1_nsi	r/w	1'b1	PWM channel 1 negative set idle state
6	pwm_ch1_nen	r/w	1'b0	PWM channel 1 negative enable pwm out
5	pwm_ch1_psi	r/w	1'b0	PWM channel 1 positive set idle state
4	pwm_ch1_pen	r/w	1'b0	PWM channel 1 positive enable pwm out
3	pwm_ch0_nsi	r/w	1'b1	PWM channel 0 negative set idle state
2	pwm_ch0_nen	r/w	1'b0	PWM channel 0 negative enable pwm out
1	pwm_ch0_psi	r/w	1'b0	PWM channel 0 positive set idle state
0	pwm_ch0_pen	r/w	1'b0	PWM channel 0 positive enable pwm out

12.4.4 pwm_mc0_period

Address: 0x4000a448

pwm_int_period_cnt

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

pwm_period

Bits	Name	Type	Reset	Description
31:16	pwm_int_period_cnt	r/w	16'd0	PWM interrupt period counter threshold
15:0	pwm_period	r/w	16'd0	PWM period setting

12.4.5 pwm_mc0_dead_time

Address: 0x4000a44c

pwm_ch3_dtg

pwm_ch2_dtg

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

pwm_ch1_dtg

pwm_ch0_dtg

Bits	Name	Type	Reset	Description
31:24	pwm_ch3_dtg	r/w	8'h0	PWM Channel 3 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)

Bits	Name	Type	Reset	Description
23:16	pwm_ch2_dtg	r/w	8'h0	PWM Channel 2 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)
15:8	pwm_ch1_dtg	r/w	8'h0	PWM Channel 1 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)
7:0	pwm_ch0_dtg	r/w	8'h0	PWM Channel 0 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)

12.4.6 pwm_mc0_ch0_thre

Address: 0x4000a450

pwm_ch0_threH

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

pwm_ch0_threL

Bits	Name	Type	Reset	Description
31:16	pwm_ch0_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch0_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

12.4.7 pwm_mc0_ch1_thre

Address: 0x4000a454

pwm_ch1_threH

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

pwm_ch1_threL

Bits	Name	Type	Reset	Description
31:16	pwm_ch1_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch1_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

12.4.8 pwm_mc0_ch2_thre

Address: 0x4000a458

pwm_ch2_threH

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

pwm_ch2_threL

Bits	Name	Type	Reset	Description
31:16	pwm_ch2_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch2_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

12.4.9 pwm_mc0_ch3_thre

Address: 0x4000a45c

pwm_ch3_threH

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

pwm_ch3_threL

Bits	Name	Type	Reset	Description
31:16	pwm_ch3_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller than threL
15:0	pwm_ch3_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger than threH

12.4.10 pwm_mc0_int_sts

Address: 0x4000a460

RSVD															
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

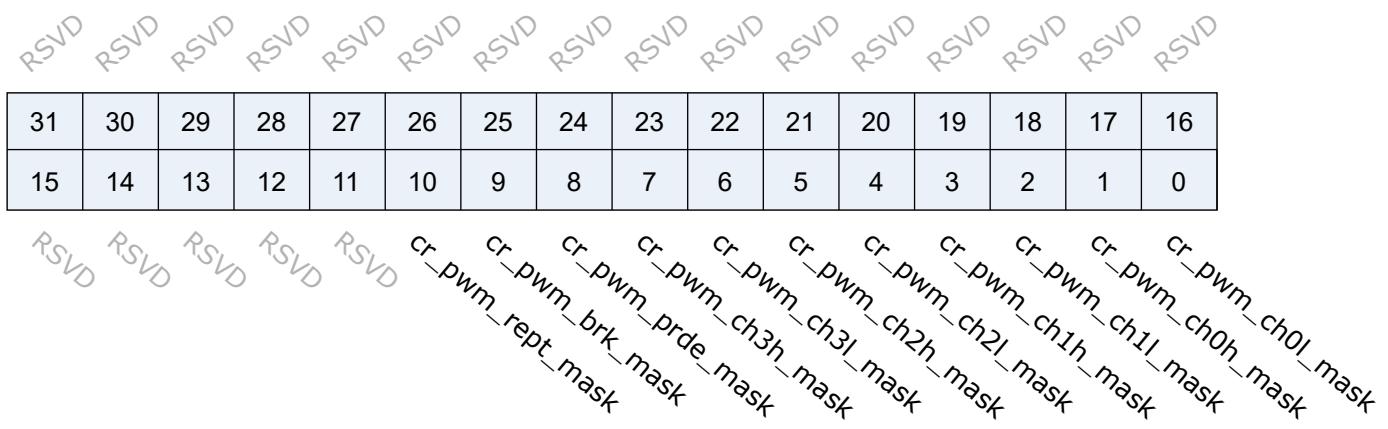
RSVD RSVD

Bits	Name	Type	Reset	Description
31:11	RSVD			
10	pwm_rept_int	r	1'b0	PWM repeat count interrupt status
9	pwm_brk_int	r	1'b0	PWM break interrupt status, triggered when ext_break is asserted Note: pwm_sw_break_en will NOT trigger this interrupt
8	pwm_prde_int	r	1'b0	PWM period end interrupt status
7	pwm_ch3h_int	r	1'b0	PWM Channel 3 ThresholdH interrupt status
6	pwm_ch3l_int	r	1'b0	PWM Channel 3 ThresholdL interrupt status
5	pwm_ch2h_int	r	1'b0	PWM Channel 2 ThresholdH interrupt status
4	pwm_ch2l_int	r	1'b0	PWM Channel 2 ThresholdL interrupt status

Bits	Name	Type	Reset	Description
3	pwm_ch1h_int	r	1'b0	PWM Channel 1 ThresholdH interrupt status
2	pwm_ch1l_int	r	1'b0	PWM Channel 1 ThresholdL interrupt status
1	pwm_ch0h_int	r	1'b0	PWM Channel 0 ThresholdH interrupt status
0	pwm_ch0l_int	r	1'b0	PWM Channel 0 ThresholdL interrupt status

12.4.11 pwm_mc0_int_mask

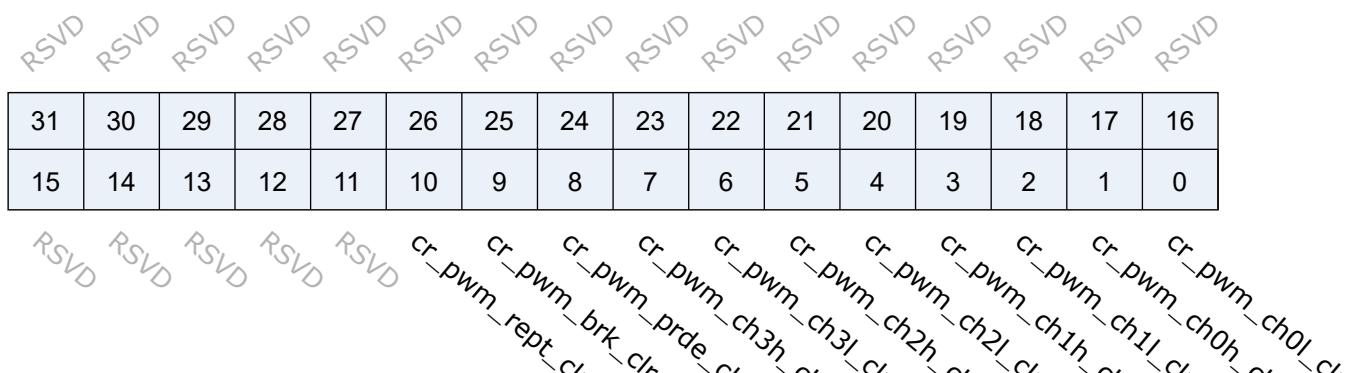
Address: 0x4000a464



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_mask	r/w	1'b1	Interrupt mask of pwm_rept_int
9	cr_pwm_brk_mask	r/w	1'b1	Interrupt mask of pwm_brk_int
8	cr_pwm_prde_mask	r/w	1'b1	Interrupt mask of pwm_prde_int
7	cr_pwm_ch3h_mask	r/w	1'b1	Interrupt mask of pwm_ch3h_int
6	cr_pwm_ch3l_mask	r/w	1'b1	Interrupt mask of pwm_ch3l_int
5	cr_pwm_ch2h_mask	r/w	1'b1	Interrupt mask of pwm_ch2h_int
4	cr_pwm_ch2l_mask	r/w	1'b1	Interrupt mask of pwm_ch2l_int
3	cr_pwm_ch1h_mask	r/w	1'b1	Interrupt mask of pwm_ch1h_int
2	cr_pwm_ch1l_mask	r/w	1'b1	Interrupt mask of pwm_ch1l_int
1	cr_pwm_ch0h_mask	r/w	1'b1	Interrupt mask of pwm_ch0h_int
0	cr_pwm_ch0l_mask	r/w	1'b1	Interrupt mask of pwm_ch0l_int

12.4.12 pwm_mc0_int_clear

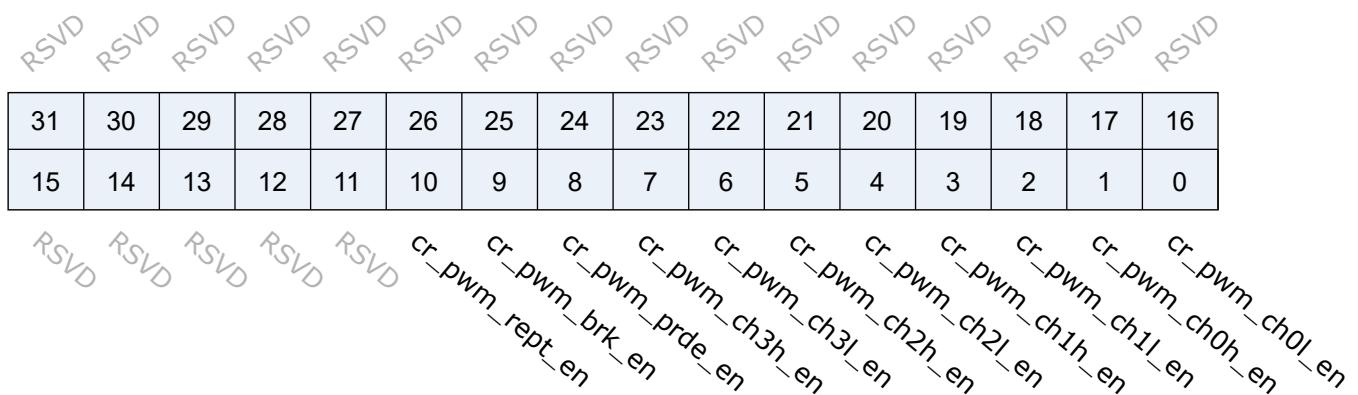
Address: 0x4000a468



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_clr	w1c	1'b0	Interrupt clear of pwm_rept_int
9	cr_pwm_brk_clr	w1c	1'b0	Interrupt clear of pwm_brk_int
8	cr_pwm_prde_clr	w1c	1'b0	Interrupt clear of pwm_prde_int
7	cr_pwm_ch3h_clr	w1c	1'b0	Interrupt clear of pwm_ch3h_int
6	cr_pwm_ch3l_clr	w1c	1'b0	Interrupt clear of pwm_ch3l_int
5	cr_pwm_ch2h_clr	w1c	1'b0	Interrupt clear of pwm_ch2h_int
4	cr_pwm_ch2l_clr	w1c	1'b0	Interrupt clear of pwm_ch2l_int
3	cr_pwm_ch1h_clr	w1c	1'b0	Interrupt clear of pwm_ch1h_int
2	cr_pwm_ch1l_clr	w1c	1'b0	Interrupt clear of pwm_ch1l_int
1	cr_pwm_ch0h_clr	w1c	1'b0	Interrupt clear of pwm_ch0h_int
0	cr_pwm_ch0l_clr	w1c	1'b0	Interrupt clear of pwm_ch0l_int

12.4.13 pwm_mc0_int_en

Address: 0x4000a46c



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_en	r/w	1'b1	Interrupt enable of pwm_rept_int
9	cr_pwm_brk_en	r/w	1'b1	Interrupt enable of pwm_brk_int
8	cr_pwm_prde_en	r/w	1'b1	Interrupt enable of pwm_prde_int
7	cr_pwm_ch3h_en	r/w	1'b1	Interrupt enable of pwm_ch3h_int
6	cr_pwm_ch3l_en	r/w	1'b1	Interrupt enable of pwm_ch3l_int
5	cr_pwm_ch2h_en	r/w	1'b1	Interrupt enable of pwm_ch2h_int
4	cr_pwm_ch2l_en	r/w	1'b1	Interrupt enable of pwm_ch2l_int
3	cr_pwm_ch1h_en	r/w	1'b1	Interrupt enable of pwm_ch1h_int
2	cr_pwm_ch1l_en	r/w	1'b1	Interrupt enable of pwm_ch1l_int
1	cr_pwm_ch0h_en	r/w	1'b1	Interrupt enable of pwm_ch0h_int
0	cr_pwm_ch0l_en	r/w	1'b1	Interrupt enable of pwm_ch0l_int

13.1 Overview

Two 32-bit counters are built in the chip, and each can independently control and configure its parameters and clock frequency.

There is one watchdog counter in the chip. Unpredictable software or hardware behavior may cause the application to malfunction, and the watchdog timer can help recover the system. If the current stage exceeds the preset time, but the watchdog is not reset or turned off, the interrupt or system reset can be triggered as configured.

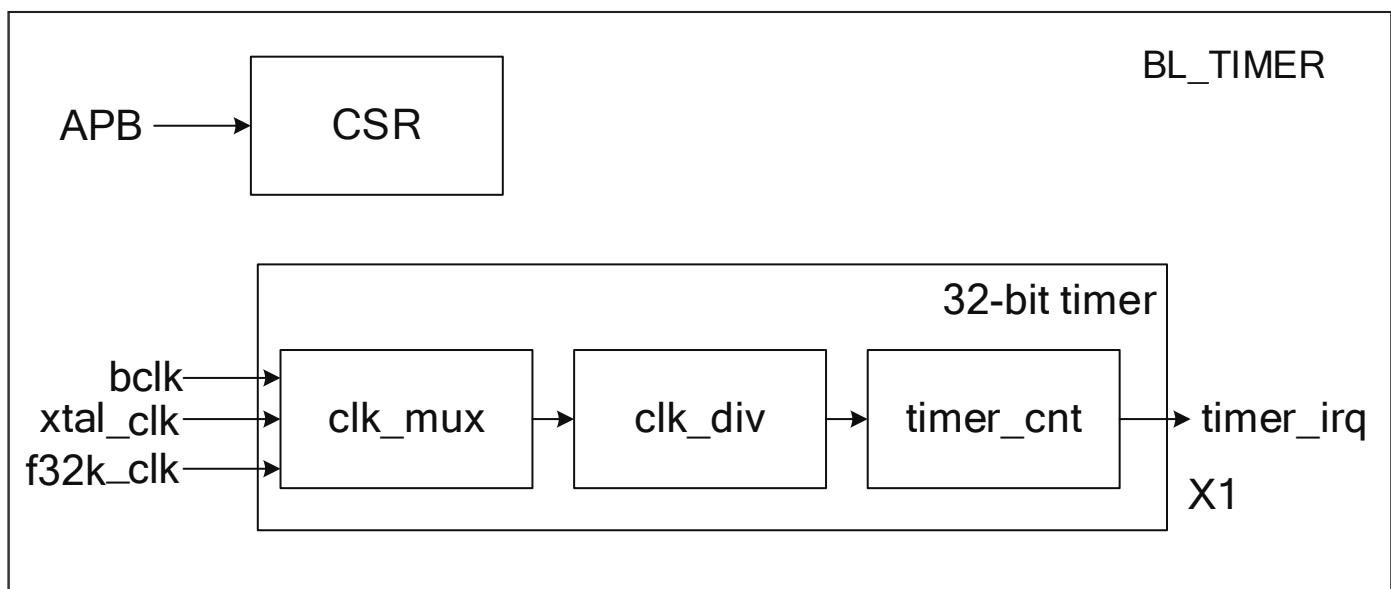


Fig. 13.1: Block diagram of timer

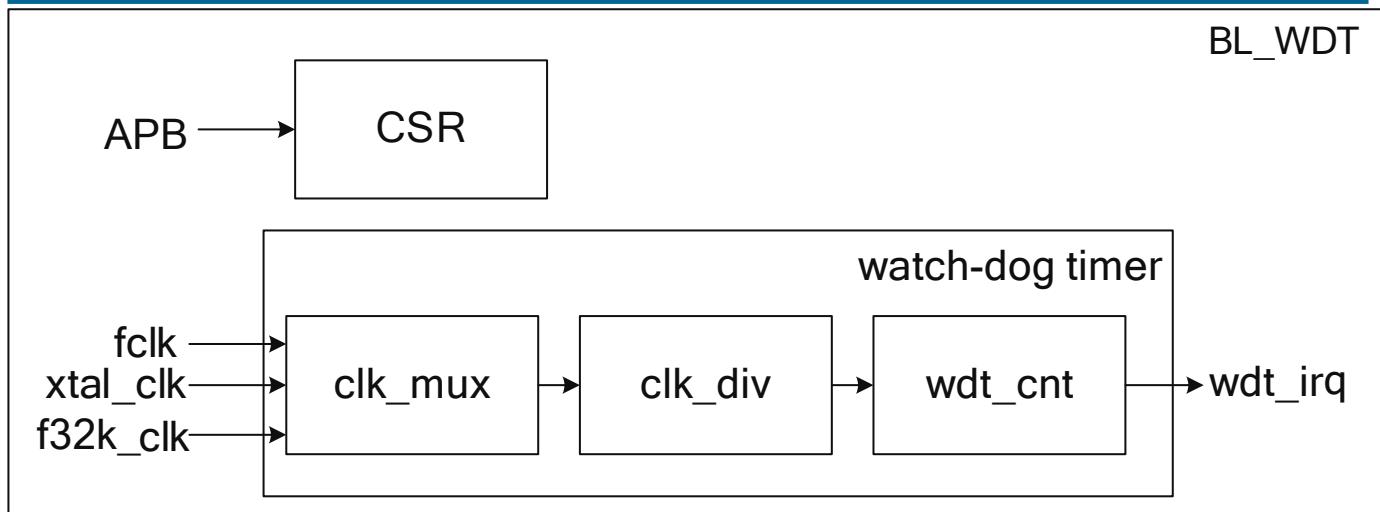


Fig. 13.2: Block diagram of watchdog timer

13.2 Features

- Multiple clock sources, up to 80M clock supported
- Bit clock divider with a division factor of 1-256
- Two 32-bit timers: channel 0 and channel 1
- Each timer has three alarm value settings, and the alarm when each set of alarm values overflows can be independently set.
- Supports Free Run mode and Pre_load mode
- With 16-bit watchdog timer
- Supports write password protection to prevent system error caused by wrong settings
- Supports two watchdog overflow modes: interrupt or reset
- Supports measuring the pulse width of external GPIO

13.3 Functional Description

There are 5 types of watchdog timer clocks:

- BCLK—bus clock
- 32K-32K clock
- 1K-1K clock
- XTAL—external crystal oscillator
- GPIO—external GPIO

It is configured by cs_wdt in the register TCCR.

There are 5 types of timer clock sources:

- BCLK-bus clock
- 32K-32K clock
- 1K-1K clock(32K frequency division)
- XTAL-external crystal oscillator
- GPIO-external GPIO

It is configured by cs_2 and cs_3 in the register TCCR.

Each counter has its own 8-bit frequency divider, which can divide the clock by 1-256. Specifically, when it is set to 0, it means no frequency division. When it is set to 1, it will divide the clock by 2, and so on. The maximum division factor is 256, and the counter will take the divided clock as the counting cycle unit.

It is configured by tcdr2, tcdrl3, and wcdrl in the register TCDR.

13.3.1 Working Principle of General Purpose Timer

Each general purpose timer contains three comparators, one counter, and one PreLoad register. When the clock source is set and the timer is started, the counter starts to count up cumulatively. When the value of counter is equal to that of the comparator, the comparison flag is set and a comparison interrupt can be generated.

You can configure the value of channel 0 comparator 0 by setting tclr2_0, that of channel 0 comparator 1 by setting tclr2_1, and that of channel 0 comparator 2 by setting tclr2_2 in the register TICR2. The tplvr2 in the register TPLVR2 sets the channel 0 preload value.

You can configure the value of channel 1 comparator 0 by setting tclr3_0, that of channel 1 comparator 1 by setting tclr3_1, and that of channel 1 comparator 2 by setting tclr3_2 in the register TICR3. The tplvr3 in the register TPLVR3 sets the channel 1 preload value.

The counter's initial value depends on the timing mode. In the FreeRun mode, this initial value is 0, and then the counter counts up cumulatively. After reaching the maximum value, it starts counting again from 0.

In the PreLoad mode, this initial value is the value of the PreLoad register, and then counts up cumulatively. When the PreLoad condition is met, the counter's value is set to the value of the PreLoad register, and then the counter starts counting up cumulatively again. In the counting process of the timer's counter, once the counter's value is consistent with one comparison value of three comparators, the comparison flag of the comparator will be set, and the corresponding comparison interrupt can be generated.

You can configure the counting mode of channel 0 by setting timer2_mode and that of channel 1 by setting timer3_mode in the register TCMR.

If the value of the PreLoad register is 10, and the values of comparators 0, 1, and 2 are 13, 16, and 19 respectively,

the working sequence of the timer in the PreLoad mode is as follows:

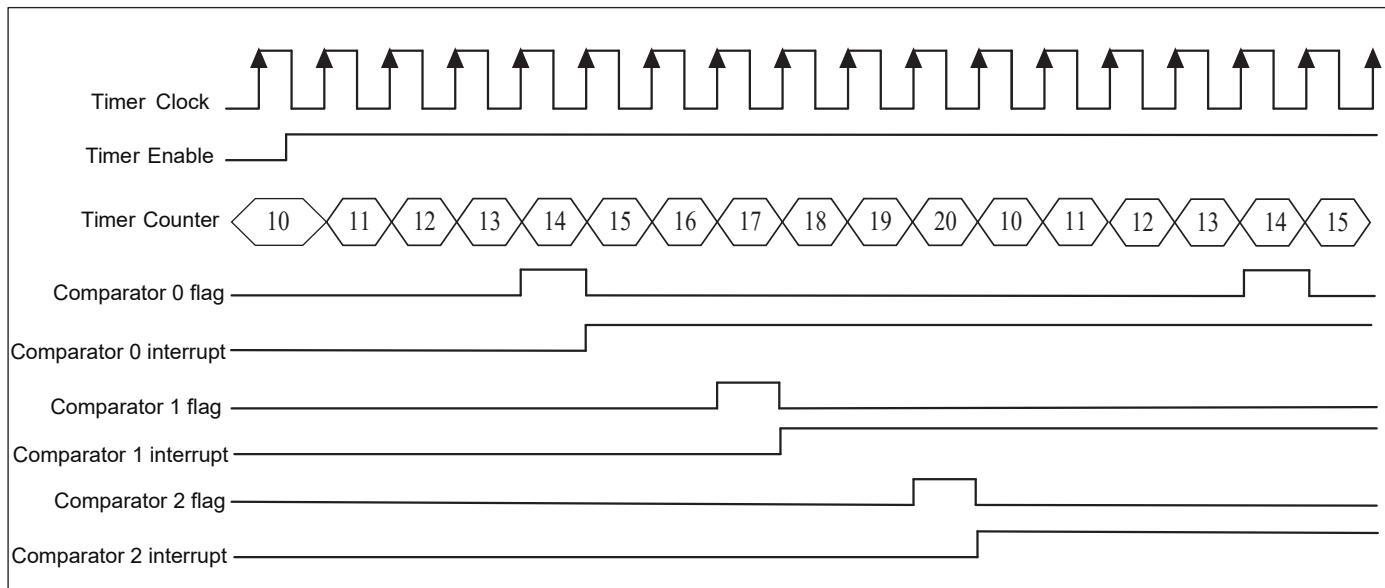


Fig. 13.3: Working sequence of timer in preLoad mode

In FreeRun mode, the working timing of the timer is basically the same as that of PreLoad, except that the counter will accumulate from 0 to the maximum value, and the mechanism of comparison flag and comparison interrupt generated during this period is the same as that of FreeRun mode.

channel 0 can use the internal clock source to calculate the pulse width of the external gpio. This function is enabled by setting `timer2_gpio_en` in the register `GPIO`. By setting the `timer2_gpio_inv` bit, it is judged whether the high level or low level of the external gpio is obtained. If the bit is 0, it means high level; if the bit is 1, it means Low level; in addition, the external gpio function needs to be set to the `gpio_tmr_clk` function. By configuring the `gpio_tmr_clk_sel[13:12]` bits in the register `dig_clk_cfg2` in the GLB module; at the same time, you need to configure a bit in the register `dig_clk_cfg2[11:8]` to 0, which needs to be used in conjunction with `gpio_tmr_clk_sel`. details as follows:

- If `gpio_tmr_clk_sel[13:12]` is configured as 0, then `chip_clk_out_0_en` in register `dig_clk_cfg2` is set to 0
- If `gpio_tmr_clk_sel[13:12]` is configured as 1, then `chip_clk_out_1_en` in register `dig_clk_cfg2` is set to 0
- If `gpio_tmr_clk_sel[13:12]` is configured as 2, then `chip_clk_out_2_en` in register `dig_clk_cfg2` is set to 0
- If `gpio_tmr_clk_sel[13:12]` is configured as 3, then `chip_clk_out_3_en` in register `dig_clk_cfg2` is set to 0

After the configuration is complete, enable the timer. When the `gpio_lat_ok` in the register `GPIO` is set to 1, the values of the register `GPIO_LAT2` and the register `GPIO_LAT1` are obtained. The calculation method of the pulse width of the external gpio: $(\text{GPIO_LAT2}-\text{GPIO_LAT1}) * \text{the width of 1 cycle of the internal clock source of the timer}$.

For example: the internal clock source of the timer is 80M, the frequency of the external gpio is 2M, and the duty ratio is 1:1. Write 1 to the `timer2_gpio_inv` bit to calculate the width of the low level of the external gpio. After the above configuration is completed, the difference between the register `GPIO_LAT2` and the register `GPIO_LAT1` is 20, then the low level width of the external gpio is: $20 * (1 / 80000000) = 1 / 4000000$; Write 0 to the `timer2_gpio_inv` bit,

which means to calculate the width of the high level of the external gpio. After the above configuration is completed, the difference between the register GPIO_LAT2 and the register GPIO_LAT1 is 20, then the high level width of the external gpio is: $20 * (1 / 80000000) = 1 / 4000000$;

13.3.2 Working Principle of Watchdog Timer

The watchdog timer integrates a counter and a comparator. The counter counts up from 0 cumulatively. If the counter is reset, it counts up again from 0. When the value of counter is equal to that of the comparator, it can generate a comparison interrupt signal or a system reset signal. Users may use one of them as required. The watchdog counter will add 1 to each counting cycle unit, and the software can reset this counter to zero through APB at any time.

The wmr in the register WMR sets the comparison value.

If the comparator value is 6, the working sequence of Watchdog is shown as follows:

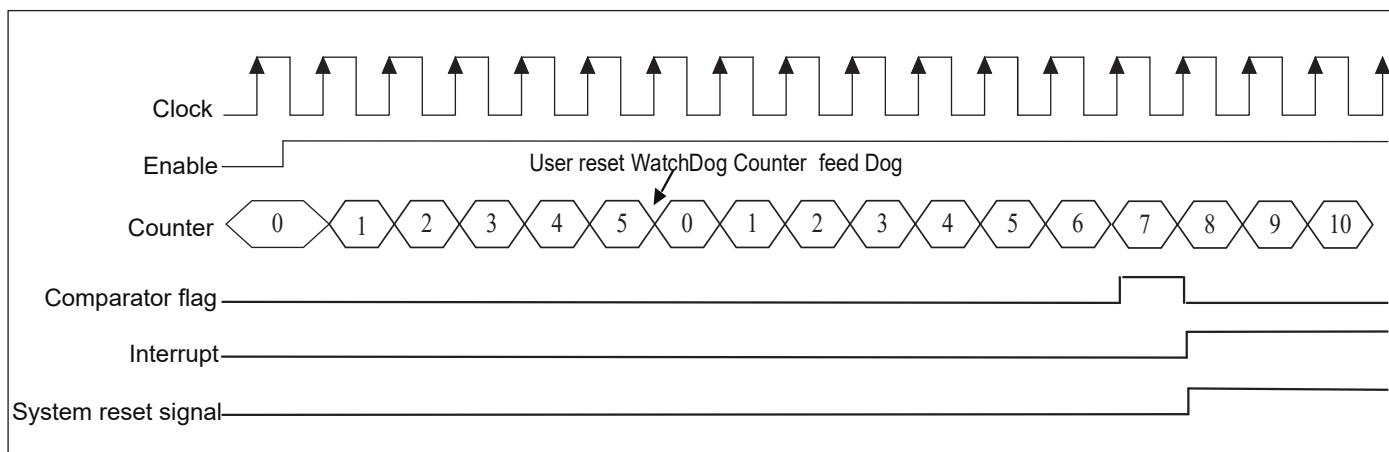


Fig. 13.4: Working sequence of watchdog

13.3.3 Alarm Setting

Each set of counter has three comparison values to provide software settings, and it can set whether each comparison value triggers an alarm interrupt. When the counter's value matches the comparison value and an alarm will be given, the counter will notify the processor through interrupts.

Through APB, the software can read whether there is an alarm at present and which comparison value triggers the alarm interrupt. When the alarm interrupt is cleared, the alarm state will also be cleared synchronously.

13.3.4 Watchdog Alarm

Each counter can be configured with one comparison value. When the watchdog counter is too late to be reset to zero due to a system error, which causes the watchdog counter to exceed the comparison value, it will trigger the watchdog alarm. There are two alarm modes. One is to notify the software to process it by generating an interrupt. The other one is to perform watchdog reset. When the watchdog reset is triggered, it will notify the system's reset controller and prepare for system reset. When everything is ready, the watchdog reset will be performed. It is worth noting that the software can read WSR register through APB to know whether watchdog reset has occurred.

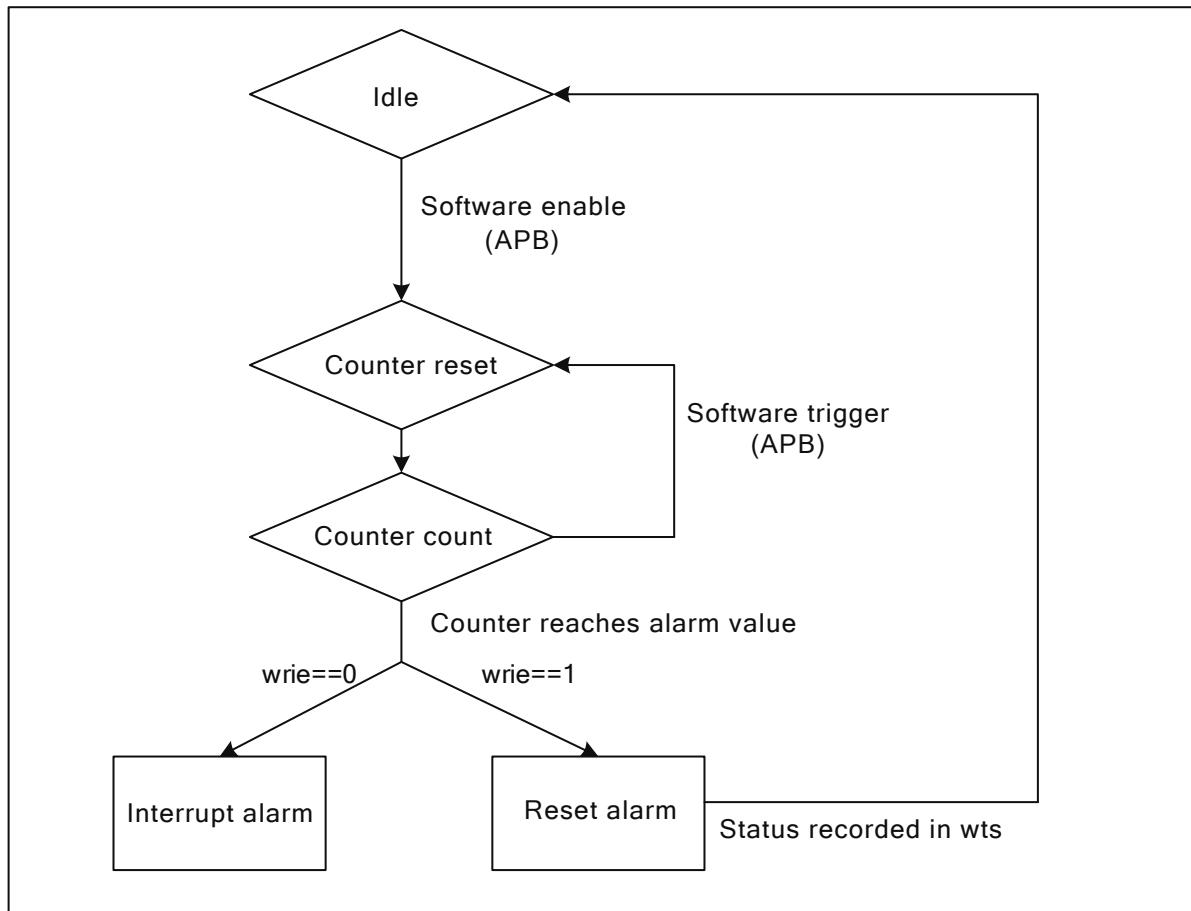


Fig. 13.5: Watchdog alarm mechanism

13.4 Register description

Name	Description
TCCR	Timer Clock Source
TMR2_0	Timer2 Match Value 0
TMR2_1	Timer2 Match Value 1
TMR2_2	Timer2 Match Value 2

Name	Description
TMR3_0	Timer3 Match Value 0
TMR3_1	Timer3 Match Value 1
TMR3_2	Timer3 Match Value 2
TCR2	Timer2 Counter Value
TCR3	Timer3 Counter Value
TSR2	Timer2 Match Status
TSR3	Timer3 Match Status
TIER2	Timer2 Match Interrupt Enable
TIER3	Timer3 Match Interrupt Enable
TPLVR2	Timer2 Pre-Load Value
TPLVR3	Timer3 Pre-Load Value
TPLCR2	Timer2 Pre-Load Control
TPLCR3	Timer3 Pre-Load Control
WMER	Watch-dog reset/interrupt Mode
WMR	Watch-dog Match Value
WVR	Watch-dog Counter Value
WSR	Watch-dog Reset Status
TICR2	Timer2 Interrupt Clear
TICR3	Timer3 Interrupt Clear
WICR	WDT Interrupt Clear
TCER	Timer Counter Enable/Clear
TCMR	Timer Counter Mode
TILR2	Timer2 Match Interrupt Mode
TILR3	Timer3 Match Interrupt Mode
WCR	WDT Counter Reset
WFAR	WDT Access Key1
WSAR	WDT Access Key2
TCVWR2	Timer2 Counter Latch Value
TCVWR3	Timer3 Counter Latch Value

Name	Description
TCVSYN2	Timer2 Counter Sync Value
TCVSYN3	Timer3 Counter Sync Value
TCDR	Timer Division
GPIO	GPIO Mode
GPIO_LAT1	GPIO Latch Value1
GPIO_LAT2	GPIO Latch Value2

13.4.1 TCCR

Address: 0x4000a500

ID																tmr_rsv							
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						

RSVD *RSVD* *RSVD* *RSVD* *cs_wdt* *cs_3* *cs_2*

Bits	Name	Type	Reset	Description
31:24	ID	r	8'ha5	
23:16	tmr_rsv	rsvd	0	
15:12	RSVD			
11:8	cs_wdt	r/w	4'd1	WDT 0:fclk / 1:f32k / 2:1k / 3:32M / 4:GPIO / 5:No clock
7:4	cs_3	r/w	4'd5	Timer3 0:fclk / 1:f32k / 2:1k / 3:32M / 4:GPIO / 5:No clock
3:0	cs_2	r/w	4'd5	Timer2 0:fclk / 1:f32k / 2:1k / 3:32M / 4:GPIO / 5:No clock

13.4.2 TMR2_0

Address: 0x4000a510

tmr2_0															
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

tmr2_0

Bits	Name	Type	Reset	Description
31:0	tmr2_0	r/w	32'hfffffff	Timer2 Match Value 0

13.4.3 TMR2_1

Address: 0x4000a514

tmr2_1

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

tmr2_1

Bits	Name	Type	Reset	Description
31:0	tmr2_1	r/w	32'hfffffff	Timer2 Match Value 1

13.4.4 TMR2_2

Address: 0x4000a518

tmr2_2

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

tmr2_2

Bits	Name	Type	Reset	Description
31:0	tmr2_2	r/w	32'hfffffff	Timer2 Match Value 2

13.4.5 TMR3_0

Address: 0x4000a51c

tmr3_0

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

tmr3_0

Bits	Name	Type	Reset	Description
31:0	tmr3_0	r/w	32'hfffffff	Timer3 Match Value 0

13.4.6 TMR3_1

Address: 0x4000a520

tmr3_1

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

tmr3_1

Bits	Name	Type	Reset	Description
31:0	tmr3_1	r/w	32'hfffffff	Timer3 Match Value 1

13.4.7 TMR3_2

Address: 0x4000a524

tmr3_2

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

tmr3_2

Bits	Name	Type	Reset	Description
31:0	tmr3_2	r/w	32'hfffffff	Timer3 Match Value 2

13.4.8 TCR2

Address: 0x4000a52c

tcr2_cnt

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

tcr2_cnt

Bits	Name	Type	Reset	Description
31:0	tcr2_cnt	r	0	Timer2 Counter Value

13.4.9 TCR3

Address: 0x4000a530

tcr3_cnt

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

tcr3_cnt

Bits	Name	Type	Reset	Description
31:0	tcr3_cnt	r	0	Timer3 Counter Value

13.4.10 TSR2

Address: 0x4000a538

RSVD																
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	

RSVD RSVD

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tsr2_2	r	0	Timer2 match value 2 status/Clear interrupt would also clear this bit
1	tsr2_1	r	0	Timer2 match value 1 status/Clear interrupt would also clear this bit
0	tsr2_0	r	0	Timer2 match value 0 status/Clear interrupt would also clear this bit

13.4.11 TSR3

Address: 0x4000a53c

RSVD																							
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								

rsrvd rsrvd

tsr3_2 *tsr3_1* *tsr3_0*

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tsr3_2	r	0	Timer3 match value 2 status/Clear interrupt would also clear this bit
1	tsr3_1	r	0	Timer3 match value 1 status/Clear interrupt would also clear this bit
0	tsr3_0	r	0	Timer3 match value 0 status/Clear interrupt would also clear this bit

13.4.12 TIER2

Address: 0x4000a544

RSVD																							
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								

rsrvd rsrvd

tier2_2 *tier2_1* *tier2_0*

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tier2_2	r/w	0	Timer2 match value 2 interrupt enable
1	tier2_1	r/w	0	Timer2 match value 1 interrupt enable
0	tier2_0	r/w	0	Timer2 match value 0 interrupt enable

13.4.13 TIER3

Address: 0x4000a548

RSVD																							
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								

RSVD tier3_2 tier3_1 tier3_0

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tier3_2	r/w	0	Timer3 match value 2 interrupt enable
1	tier3_1	r/w	0	Timer3 match value 1 interrupt enable
0	tier3_0	r/w	0	Timer3 match value 0 interrupt enable

13.4.14 TPLVR2

Address: 0x4000a550

tplvr2

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

tplvr2

Bits	Name	Type	Reset	Description
31:0	tplvr2	r/w	0	Timer2 Pre-Load Value

13.4.15 TPLVR3

Address: 0x4000a554

tplvr3

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

tplvr3

Bits	Name	Type	Reset	Description
31:0	tplvr3	r/w	0	Timer3 Pre-Load Value

13.4.16 TPLCR2

Address: 0x4000a55c

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | |

Bits	Name	Type	Reset	Description
31:2	RSVD			
1:0	tplcr2	r/w	0	Timer2 pre-load control 2'd0 - No pre-load 2'd1 - Pre-load with match comparator 0 2'd2 - Pre-load with match comparator 1 2'd3 - Pre-load with match comparator 2

13.4.17 TPLCR3

Address: 0x4000a560

Bits	Name	Type	Reset	Description
31:2	RSVD			

Bits	Name	Type	Reset	Description
1:0	tplcr3	r/w	0	Timer3 pre-load control 2'd0 - No pre-load 2'd1 - Pre-load with match comparator 0 2'd2 - Pre-load with match comparator 1 2'd3 - Pre-load with match comparator 2

13.4.18 WMER

Address: 0x4000a564

RSVD																	
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		
RSVD	wrie	we															

Bits	Name	Type	Reset	Description
31:2	RSVD			
1	wrie	r/w	0	WDT reset/interrupt mode 1'b0 - WDT expiration to generate interrupt 1'b1 - WDT expiration to generate reset source
0	we	r/w	0	WDT enable register

13.4.19 WMR

Address: 0x4000a568

RSVD	wdt_align																
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	wmr	

Bits	Name	Type	Reset	Description
31:17	RSVD			
16	wdt_align	r/w	0	WDT compare value update align interrupt
15:0	wmr	r/w	16'hffff	WDT counter match value

13.4.20 WVR

Address: 0x4000a56c

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | | | |

wdt_cnt

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	wdt_cnt	r	0	WDT counter value

13.4.21 WSR

Address: 0x4000a570

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | | | |

wts

Bits	Name	Type	Reset	Description
31:1	RSVD			
0	wts	w	0	WDT reset status Write 0 to clear the WDT reset status Read 1 indicates reset was caused by the WDT

13.4.22 TICR2

Address: 0x4000a578

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | |

RSVD tclr2_2 tclr2_1 tclr2_0

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tclr2_2	w	0	Timer2 Interrupt clear for match comparator 2
1	tclr2_1	w	0	Timer2 Interrupt clear for match comparator 1
0	tclr2_0	w	0	Timer2 Interrupt clear for match comparator 0

13.4.23 TICR3

Address: 0x4000a57c

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | |

RSVD tclr3_2 tclr3_1 tclr3_0

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tclr3_2	w	0	Timer3 Interrupt clear for match comparator 2
1	tclr3_1	w	0	Timer3 Interrupt clear for match comparator 1
0	tclr3_0	w	0	Timer3 Interrupt clear for match comparator 0

13.4.24 WICR

Address: 0x4000a580

RSVD																			
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				

wiclr

Bits	Name	Type	Reset	Description
31:1	RSVD			
0	wiclr	w	0	WDT Interrupt Clear

13.4.25 TCER

Address: 0x4000a584

RSVD																			
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				

tcr3_cnt_clr tcr2_cnt_clr timer3_en timer2_en RSVD

Bits	Name	Type	Reset	Description
31:7	RSVD			
6	tcr3_cnt_clr	r/w	0	Timer3 count clear
5	tcr2_cnt_clr	r/w	0	Timer2 count clear
4:3	RSVD			
2	timer3_en	r/w	0	Timer3 count enable
1	timer2_en	r/w	0	Timer2 count enable
0	RSVD			

13.4.26 TCMR

Address: 0x4000a588

RSVD															
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

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD timer3_align timer2_align RSVD RSVD timer3_mode timer2_mode RSVD

Bits	Name	Type	Reset	Description
31:7	RSVD			
6	timer3_align	r/w	0	Timer3 compare value update align interrupt
5	timer2_align	r/w	0	Timer2 compare value update align interrupt
4:3	RSVD			
2	timer3_mode	r/w	0	0:pre-load mode 1:free run mode
1	timer2_mode	r/w	0	0:pre-load mode 1:free run mode
0	RSVD			

13.4.27 TILR2

Address: 0x4000a590

RSVD															
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

RSVD RSVD

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tilr2_2	r/w	0	0:level 1:edge
1	tilr2_1	r/w	0	0:level 1:edge

Bits	Name	Type	Reset	Description
0	tilr2_0	r/w	0	0:level 1:edge

13.4.28 TILR3

Address: 0x4000a594

RSVD	RSVD																	
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			
																tilr3_2	tilr3_1	tilr3_0

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tilr3_2	r/w	0	0:level 1:edge
1	tilr3_1	r/w	0	0:level 1:edge
0	tilr3_0	r/w	0	0:level 1:edge

13.4.29 WCR

Address: 0x4000a598

RSVD																		
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			
																wcr		

Bits	Name	Type	Reset	Description
31:1	RSVD			
0	wcr	w	0	WDT Counter Reset

13.4.30 WFAR

Address: 0x4000a59c

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | | | |

wfar

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	wfar	w	0	WDT access key1 - 16'hBABA

13.4.31 WSAR

Address: 0x4000a5a0

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | | | |

wsar

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	wsar	w	0	WDT access key2 - 16'hEB10

13.4.32 TCVWR2

Address: 0x4000a5a8

tcr2_cnt_lat

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				

tcr2_cnt_lat

Bits	Name	Type	Reset	Description
31:0	tcr2_cnt_lat	r	0	Timer2 Counter Latch Value

13.4.33 TCVWR3

Address: 0x4000a5ac

tcr3_cnt_lat

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

tcr3_cnt_lat

Bits	Name	Type	Reset	Description
31:0	tcr3_cnt_lat	r	0	Timer3 Counter Latch Value

13.4.34 TCVSYN2

Address: 0x4000a5b4

tcr2_cnt_sync

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

tcr2_cnt_sync

Bits	Name	Type	Reset	Description
31:0	tcr2_cnt_sync	r	0	Timer2 Counter Sync Value (continue readable)

13.4.35 TCVSYN3

Address: 0x4000a5b8

tcr3_cnt_sync

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

tcr3_cnt_sync

Bits	Name	Type	Reset	Description
31:0	tcr3_cnt_sync	r	0	Timer3 Counter Sync Value (continue readable)

13.4.36 TCDR

Address: 0x4000a5bc

wcdr								tcdr3							
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

Bits	Name	Type	Reset	Description
31:24	wcdr	r/w	0	WDT clock division value register
23:16	tcdr3	r/w	0	Timer3 clock division value register
15:8	tcdr2	r/w	0	Timer2 clock division value register
7:0	RSVD			

13.4.37 GPIO

Address: 0x4000a5c0

Register Map for GPIO and Timer Functions															
GPIO Functions		RSVD		RSVD		RSVD		RSVD		RSVD		RSVD		RSVD	
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
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	wdt_gpio_inv	timer3_gpio_inv	timer2_gpio_inv	RSVD	RSVD	RSVD	RSVD	RSVD

Bits	Name	Type	Reset	Description
31	gpio_lat_ok	r	0	Latch Done. Pulse width = (GPIO_LAT2 - GPIO_LAT1) * (Timer2 Cycle)

Bits	Name	Type	Reset	Description
30:8	RSVD			
7	wdt_gpio_inv	r/w	0	WDT gpio polarity 0:pos 1:neg
6	timer3_gpio_inv	r/w	0	Timer3 gpio polarity 0:pos 1:neg
5	timer2_gpio_inv	r/w	0	Timer2 gpio polarity 0:pos 1:neg
4:2	RSVD			
1	timer2_gpio_en	r/w	0	Timer2 gpio measure enable
0	RSVD			

13.4.38 GPIO_LAT1

Address: 0x4000a5c4

gpio_lat1

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

gpio_lat1

Bits	Name	Type	Reset	Description
31:0	gpio_lat1	r	0	Pos-Edge Latch Timer2

13.4.39 GPIO_LAT2

Address: 0x4000a5c8

gpio_lat2

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

gpio_lat2

Bits	Name	Type	Reset	Description
31:0	gpio_lat2	r	0	Neg-Edge Latch Timer2

14.1 Overview

InterIC Sound (I2S), also Integrated Interchip Sound (IIS), is a digital audio transmission specification defined by Philips in 1986 (revised in 1996) for transmitting digital audio data between internal devices of a system.

I2S transmits clock signals and data signals separately, so that the receiver does not need to restore the clock from the data signal, reducing the design difficulty of the receiver.

14.2 Features

- Supports master and slave modes
- Supported data formats: LeftJustified/RightJustified/Normal I2S/DSP
- Supports 8/16/24/32-bit data width
- Supports data MSB/LSB switching
- Supports DMA transfer mode
- Supports 4-channel and 6-channel modes in addition to mono/2-channel mode
- Supports playing of mono audio in the 2-channel mode and channel mode swapping
- The 2-channel recording data with less than 16 bits can be merged into FIFO data with less than 32 bits
- Supports 16/32/48/64-bit frame size
- Supports dynamic mute switch function
- TX FIFO has a width of 32 bits and a depth of 16
- RX FIFO has a width of 32 bits and a depth of 16

14.3 Functional Description

Table 14.1: I2S pin list

Name	Type	Description
I2Sx_DI	Input	Serial data input
I2Sx_DO	Output	Serial data output
I2Sx_BCLK	Input/output	Synchronous transmission clock: output (master)/input (slave)
I2Sx_FS	Input/output	Data start/end signal: output (master)/input (slave)

14.4 Functional Description

14.4.1 Data Formats

I2S supports Normal I2S/LeftJustified/RightJustified modes. Normal I2S is a special case of the LeftJustified mode. The mode is set by I2S_CONFIG[17:16].

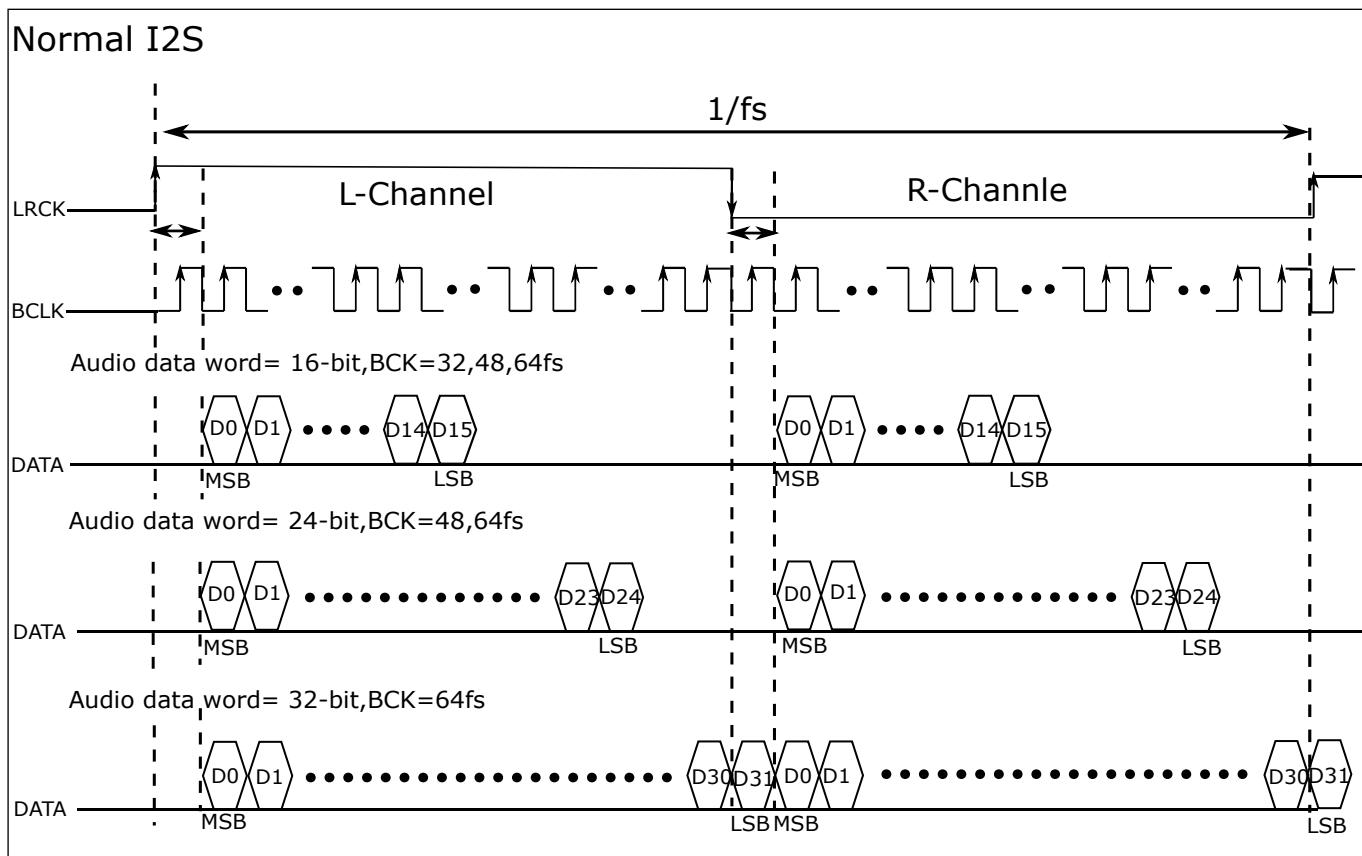
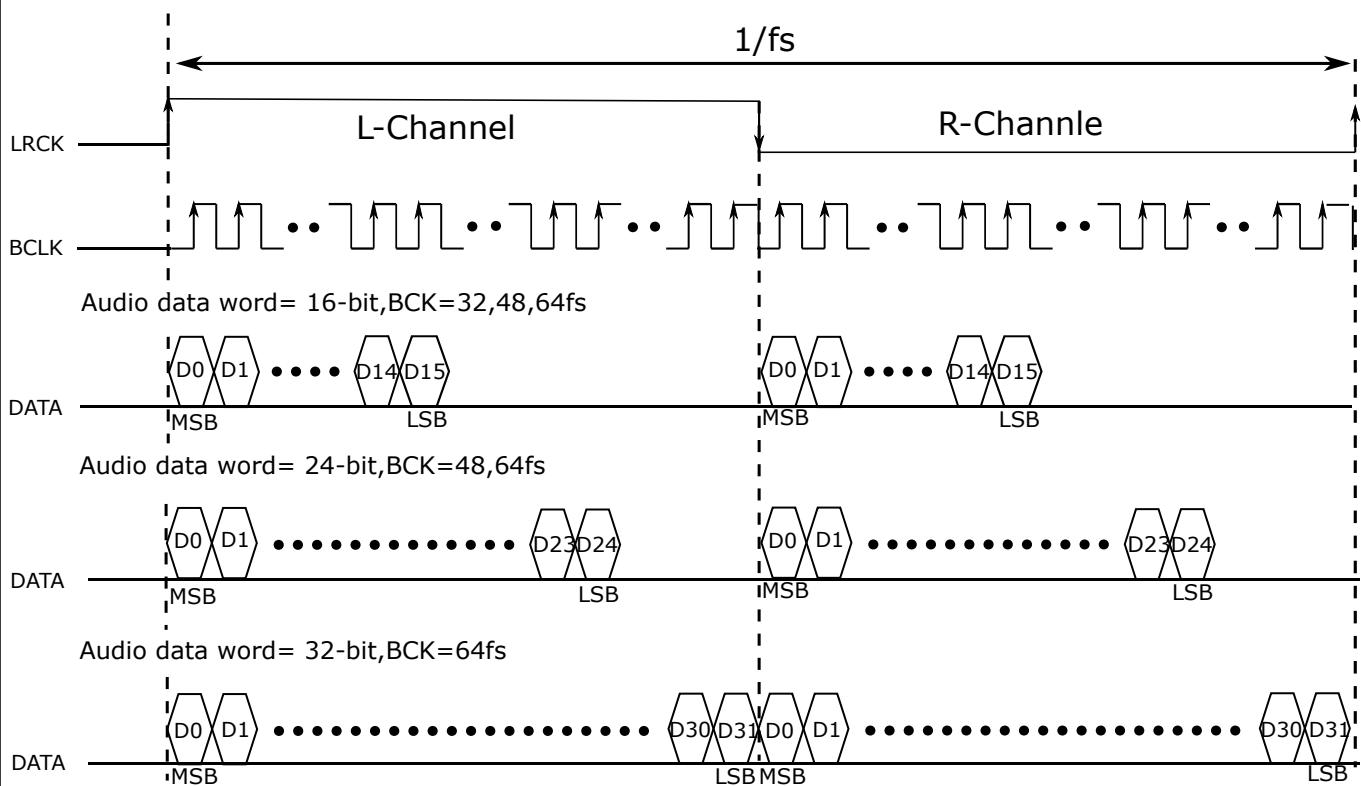


Fig. 14.1: Normal I2S

OFFSET is configured by I2S_CONFIG[25:20]. The only difference between LeftJustified and Normal I2S modes is the different configuration of I2S_CONFIG[25:20].

Left-Justified



Right-Justified

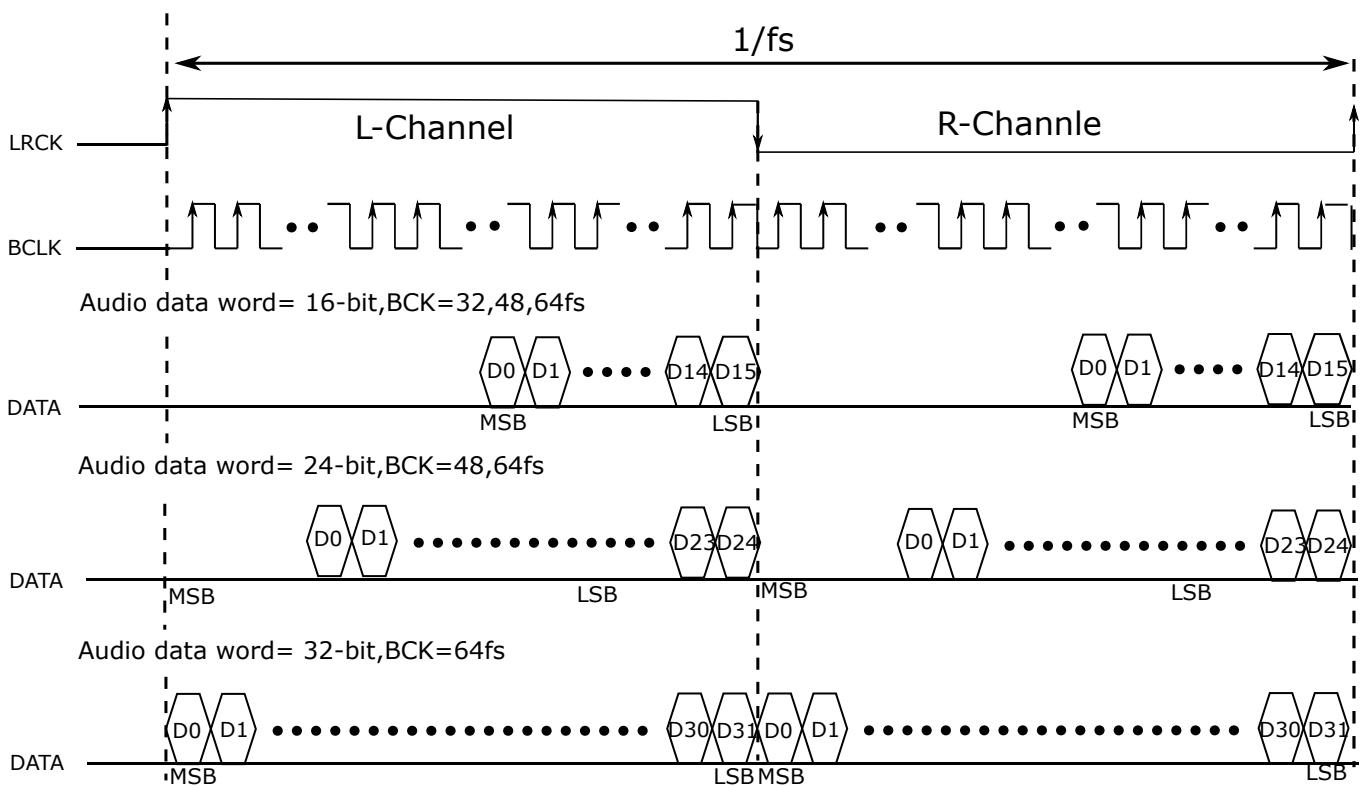


Fig. 14.2: I2S LeftJustified/RightJustified

DSP-MTK TDM64

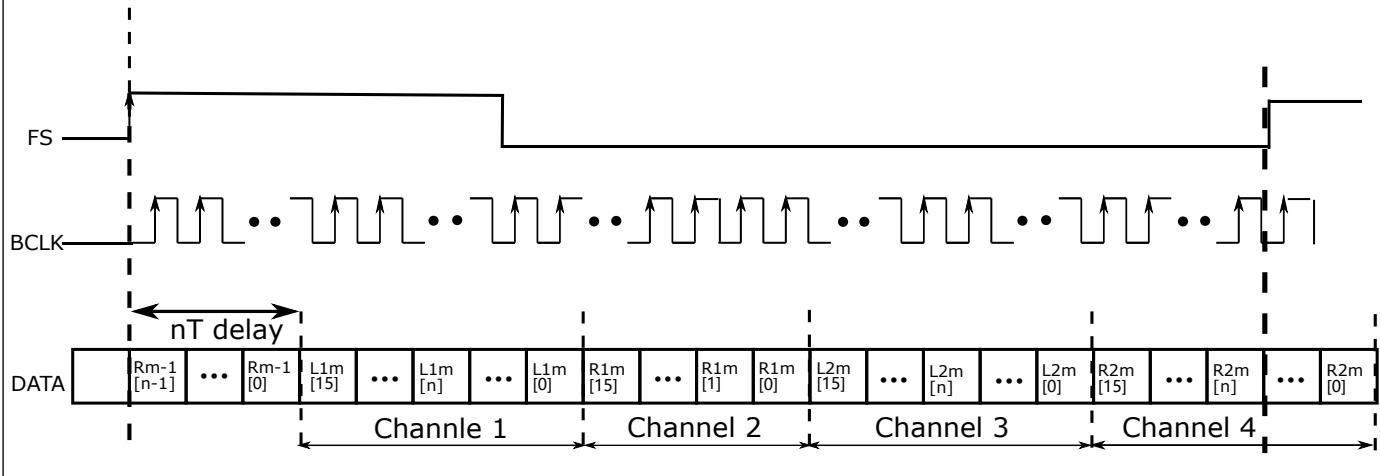


Fig. 14.3: I2S TDM64 Mode—6-Channel Recording

The width of a single pulse is controlled by I2S_CONFIG[6]. When it is set to 0, the width of high-level pulse of FS signal line is the width of data size, and when it is set to 1, that width is 1. Generally, this register is set to 1 in the multi-channel TDM64 mode.

14.4.2 Basic architecture

14.4.3 Clock source

The clock source of I2S is provided by audio PLL, and the clock divider is used to divide the clock source and then generate the clock signal to drive I2S, as shown below:

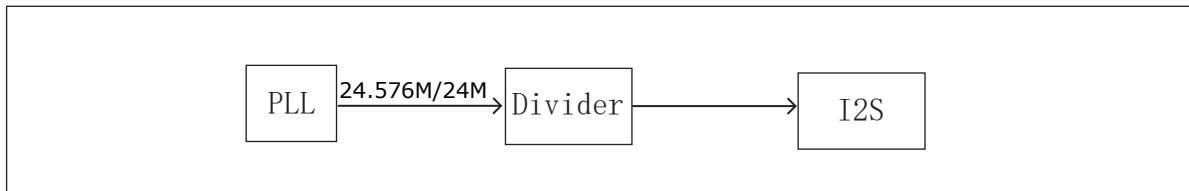


Fig. 14.4: I2S clock

14.4.4 I2S Interrupt

I2S supports the following interrupt control modes:

- TX FIFO request interrupt
- RX FIFO request interrupt
- Overrun/underrun error interrupt

A TX FIFO request interrupt will be generated when TX_FIFO_CNT in I2S_FIFO_CONFIG_1 is greater than TX_-

FIFO_TH. When the condition is not met, the interrupt flag will be cleared automatically.

A RX FIFO request interrupt will be generated when RX_FIFO_CNT in I2S_FIFO_CONFIG_1 is greater than RX_FIFO_TH. When the condition is not met, the interrupt flag will be cleared automatically.

If the TX/RX FIFO overflows or underflows, it will trigger the error interrupt. When the error disappears, the flag bit will be cleared automatically.

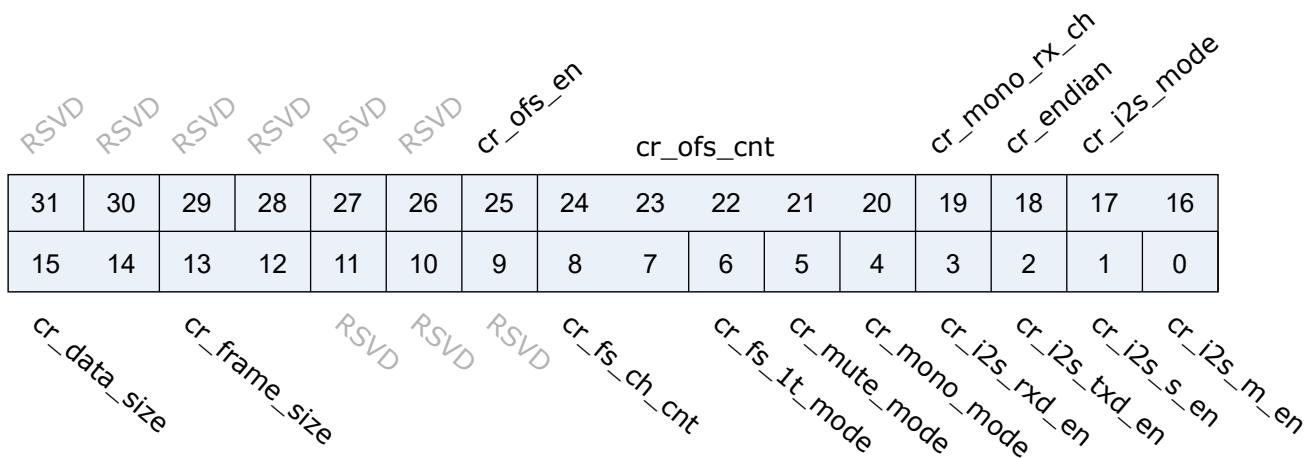
All the interrupt enable bits and interrupt flag bits of I2S are in the I2S_INT_STS register.

14.5 Register description

Name	Description
i2s_config	
i2s_int_sts	
i2s_bclk_config	
i2s_fifo_config_0	
i2s_fifo_config_1	
i2s_fifo_wdata	
i2s_fifo_rdata	
i2s_io_config	

14.5.1 i2s_config

Address: 0x40017000

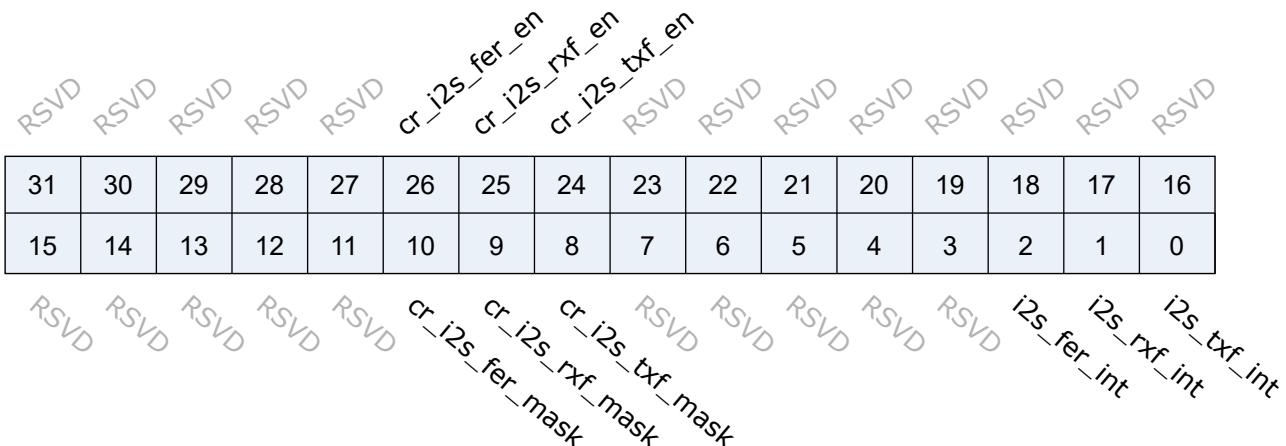


Bits	Name	Type	Reset	Description
31:26	RSVD			
25	cr_ofs_en	r/w	1'b0	Offset enable 1'b0: Disabled, 1'b1: Enabled
24:20	cr_ofs_cnt	r/w	5'd0	Offset cycle count (unit: cycle of I2S BCLK) 5'd0: 1 cycle 5'd1: 2 cycles ...
19	cr_mono_rx_ch	r/w	1'b0	RX mono mode channel select signal 1'b0: L-channel 1'b1: R-channel
18	cr_endian	r/w	1'b0	Data endian (bit reverse) 1'b0: MSB goes out first, 1'b1: LSB goes out first
17:16	cr_i2s_mode	r/w	2'd0	2'd0: Left-Justified, 2'd1: Right-Justified, 2'd2: DSP, 2'd3: Reserved
15:14	cr_data_size	r/w	2'd1	Data bit width of each channel 2'd0: 8, 2'd1: 16, 2'd2: 24, 2'd3: 32 (bits)
13:12	cr_frame_size	r/w	2'd1	Frame size of each channel 2'd0: 8, 2'd1: 16, 2'd2: 24, 2'd3: 32 (cycles)
11:9	RSVD			
8:7	cr_fs_ch_cnt	r/w	2'd0	Channel count of each frame 2'd0: FS 2-channel mode 2'd1: FS 3-channel mode (DSP mode only) 2'd2: FS 4-channel mode (DSP mode only) 2'd3: FS 6-channel mode (DSP mode only) Note: cr_mono_mode & cr_fifo_lr_merge will be invalid in 3-channel mode Note: frame_size must equal data_size in 3/4/6-channel mode
6	cr_fs_1t_mode	r/w	1'b0	1'b0: FS high/low is even, 1'b1: FS only asserts for 1 cycle
5	cr_mute_mode	r/w	1'b0	1'b0: Normal mode, 1'b1: Mute mode
4	cr_mono_mode	r/w	1'b0	1'b0: Stereo mode, 1'b1: Mono mode Note: csr_mono_mode & csr_fifo_lr_merge should NOT be enabled at the same time
3	cr_i2s_rxd_en	r/w	1'b0	Enable signal of I2S RXD signal
2	cr_i2s_txd_en	r/w	1'b0	Enable signal of I2S TXD signal
1	cr_i2s_s_en	r/w	1'b0	Enable signal of I2S Slave function, cannot enable both csr_i2s_m_en & csr_i2s_s_en

Bits	Name	Type	Reset	Description
0	cr_i2s_m_en	r/w	1'b0	Enable signal of I2S Master function, cannot enable both csr_i2s_m_en & csr_i2s_s_en

14.5.2 i2s_int_sts

Address: 0x40017004



Bits	Name	Type	Reset	Description
31:27	RSVD			
26	cr_i2s_fer_en	r/w	1'b1	Interrupt enable of i2s_fer_int
25	cr_i2s_rxf_en	r/w	1'b1	Interrupt enable of i2s_rxf_int
24	cr_i2s_txf_en	r/w	1'b1	Interrupt enable of i2s_txf_int
23:11	RSVD			
10	cr_i2s_fer_mask	r/w	1'b1	Interrupt mask of i2s_fer_int
9	cr_i2s_rxf_mask	r/w	1'b1	Interrupt mask of i2s_rxf_int
8	cr_i2s_txf_mask	r/w	1'b1	Interrupt mask of i2s_txf_int
7:3	RSVD			
2	i2s_fer_int	r	1'b0	I2S TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
1	i2s_rxf_int	r	1'b0	I2S RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto-cleared when data is popped
0	i2s_txf_int	r	1'b1	I2S TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed

14.5.3 i2s_bclk_config

Address: 0x40017010

cr_bclk_div_h															
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

cr_bclk_div_l															
---------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Bits	Name	Type	Reset	Description
31:28	RSVD			
27:16	cr_bclk_div_h	r/w	12'd1	I2S BCLK active high period (unit: cycle of i2s_clk)
15:12	RSVD			
11:0	cr_bclk_div_l	r/w	12'd1	I2S BCLK active low period (unit: cycle of i2s_clk)

14.5.4 i2s_fifo_config_0

Address: 0x40017080

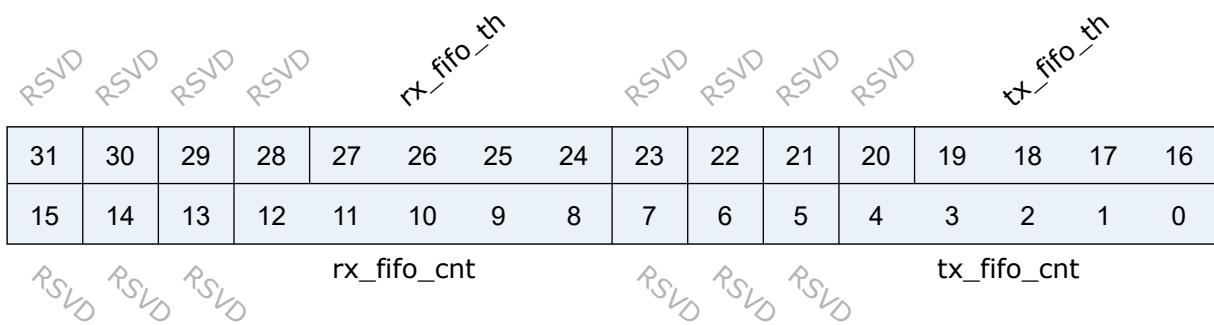
i2s_fifo_config_0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD															
cr_fifo_24b_lj cr_fifo_ir_exchg rx_fifo_underflow rx_fifo_overflow tx_fifo_clr tx_fifo_overflow rx_fifo_clr tx_fifo_en i2s_dma_tx_en															

Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_fifo_24b_lj	r/w	1'b0	FIFO 24-bit data left-justified mode 1'b0: Right-justified, 8'h0, data[23:0] 1'b1: Left-justified, data[23:0], 8'h0 Note: Valid only when cr_data_size = 2'd2 (24-bit)

Bits	Name	Type	Reset	Description
9	cr_fifo_lr_exchg	r/w	1'b0	The position of L/R channel data within each entry is exchanged if this bit is enabled Can only be enabled if data size is 8 or 16 bits and csr_fifo_lr_merge is enabled
8	cr_fifo_lr_merge	r/w	1'b0	Each FIFO entry contains both L/R channel data if this bit is enabled Can only be enabled if data size is 8 or 16 bits Note: cr_fifo_lr_merge &cr_mono_mode should NOT be enabled at the same time Note: cr_fifo_lr_merge &cr_fifo_l_shift should NOT be enabled at the same time
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	i2s_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	i2s_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

14.5.5 i2s_fifo_config_1

Address: 0x40017084



Bits	Name	Type	Reset	Description
31:28	RSVD			
27:24	rx_fifo_th	r/w	4'd0	RX FIFO threshold, dma_rx_req will not be asserted if tx_fifo_cnt is less than this value

Bits	Name	Type	Reset	Description
23:20	RSVD			
19:16	tx_fifo_th	r/w	4'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:13	RSVD			
12:8	rx_fifo_cnt	r	5'd0	RX FIFO available count
7:5	RSVD			
4:0	tx_fifo_cnt	r	5'd16	TX FIFO available count

14.5.6 i2s_fifo_wdata

Address: 0x40017088

i2s_fifo_wdata

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

i2s_fifo_wdata

Bits	Name	Type	Reset	Description
31:0	i2s_fifo_wdata	w	x	

14.5.7 i2s_fifo_rdata

Address: 0x4001708c

i2s_fifo_rdata

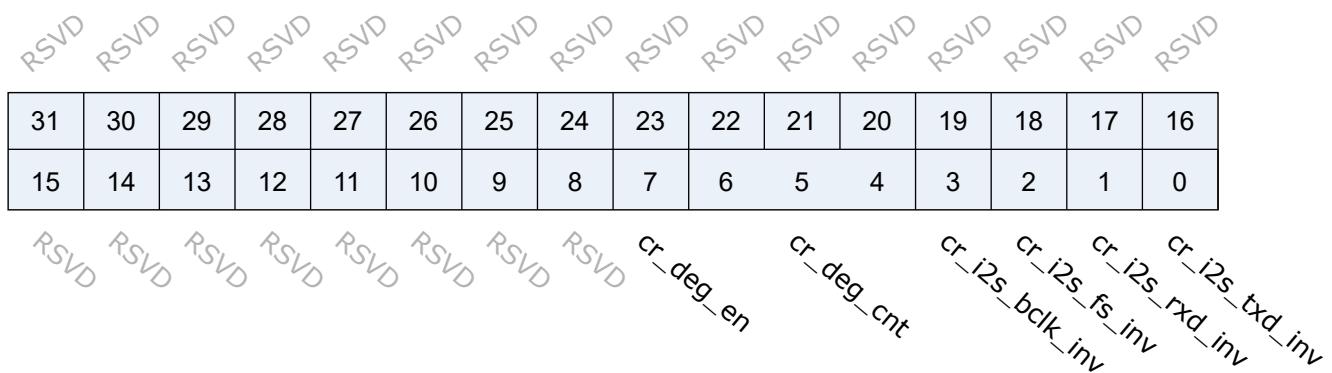
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

i2s_fifo_rdata

Bits	Name	Type	Reset	Description
31:0	i2s_fifo_rdata	r	32'h0	

14.5.8 i2s_io_config

Address: 0x400170fc



Bits	Name	Type	Reset	Description
31:8	RSVD			
7	cr_deg_en	r/w	1'b0	Deglitch enable (for all th input pins) 1'b0: Disabled, 1'b1: Enabled
6:4	cr_deg_cnt	r/w	3'd0	Deglitch cycle count (unit: cycle of I2S kernel clock) 3'd0: 1 cycle 3'd1: 2 cycles ...
3	cr_i2s_bclk_inv	r/w	1'b0	Inverse BCLK signal 0: No inverse, 1: Inverse
2	cr_i2s_fs_inv	r/w	1'b0	Inverse FS signal 0: No inverse, 1: Inverse
1	cr_i2s_rxd_inv	r/w	1'b0	Inverse RXD signal 0: No inverse, 1: Inverse
0	cr_i2s_txd_inv	r/w	1'b0	Inverse TXD signal 0: No inverse, 1: Inverse

15.1 Overview

A PWM modulation module is built in the chip to output analog signals to drive the speaker, so as to realize audio playback.

15.2 Features

- One 16-bit DAC PWM is integrated, supporting one analog PWM differential output * Sampling rate: 8k~48k * Signal to noise ratio (AW): 95 dB gain * Harmonic distortion + noise: 70dB @ 0dB gain
- Independent digital volume control that supports soft volume adjustment/mute
- 32-bit FIFO with a depth of 32
- Supports DMA transfer mode
- Interpolation filter
- Stereo data mixer selector
- Linkage with a traditional DAC

15.3 Clock Tree

The user starts the Audio PLL to select the corresponding frequency value. After one frequency division, the user enters the module and selects the division factor through the au_pwm_mode in aupwm_0. The value of this register and the frequency division ratio are shown as follows.

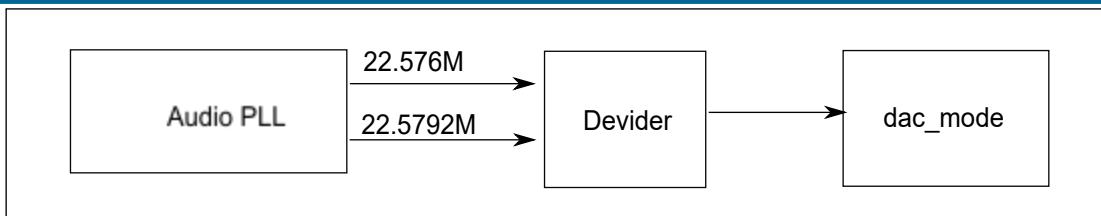


Fig. 15.1: Block Diagram of Clock

15.4 Functional Description

The block diagram of AudioPWM is shown as follows.

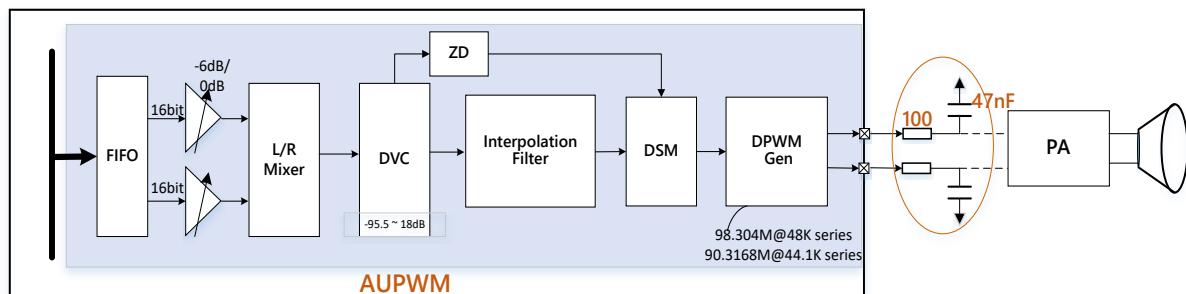


Fig. 15.2: Block Diagram of Module

AudioPWM performs DSM modulation on the data in TX FIFO after it passes the interpolation filter. Finally, it outputs a PWM signal whose duty ratio is related to TX FIFO data. Next, this signal passes through the RC low-pass filter and then can drive the speaker to play audio data.

15.4.1 AudioPWM Interrupt

AudioPWM supports the following interrupt control modes:

- TX FIFO request interrupt
- TX FIFO underrun interrupt
- TX FIFO overrun interrupt
- Volume adjustment complete interrupt

A TX FIFO request interrupt is generated when TX_DRQ_CNT in TX_FIFO_CTRL is greater than TX_TRG_LEVEL. When the condition is not met, the interrupt flag is cleared automatically.

When there is no data in TX FIFO, but the user enables TX FIFO state machine through TX_CH_EN in TX_FIFO_CTRL, the TX FIFO underrun interrupt is generated.

When the user fills in data that exceeds the maximum depth of TX FIFO, it leads to TX FIFO overflow and cause a

TX FIFO overrun interrupt.

Audio PWM volume control supports the fade-in/fade-out function. When volume adjustment is completed, the “volume adjustment complete interrupt” is generated, indicating that fade-in/fade-out adjustment is completed.

15.4.2 FIFO Format Control

AUPWM_TX_FIFO_CTRL can control the format of the audio data to be stored in FIFO.

The FIFO controller supports the following four data storage formats, which are determined by FIFO_CTRL[25:24].

- Mode 0:

DATA[15:0] = {FIFO[31:16]}

- Mode 1:

DATA[15:0] = {FIFO[23:8]}

- Mode 2:

DATA[15:0] = {FIFO[19:4]}

- Mode 3:

DATA[15:0] = {FIFO[15:0]}

Distribution of MSB

- Mode 0:

The MSB of data is 31 bits

- Mode 1:

The MSB of data is 23 bits

- Mode 2:

The MSB of data is 19 bits

- Mode 3:

The MSB of data is 15 bits

When the audio file to be played is 16-bit wide, select Mode3. The maximum resolution of DAC is 16-bit. The significance of other modes is that if the width of the audio file to be played is 32/24/20-bit, the user needs to make a trade-off, and cut out some information of low bits, to ensure that the next-stage circuit gets the 16-bit data. Here, the data of low bits is discarded by default.

15.4.3 Startup of FIFO and DMA Transfer

The data in TX FIFO of AWPWM can be transferred by DMA.

The user can obtain the current amount of valid data in FIFO in real time through the register PDM_TX_FIFO_STATUS.

The FIFO count threshold (8/16/32) for initiating DMA request is selected by configuring FIFO_CTRL[15:14], or can be determined by FIFO_CTRL[22:16].

When the count value is greater than the set threshold, and the FIFO of the channel corresponding to PDM_TX_FIFO_CTRL[12:8] is enabled, a DMA transfer is initiated.

When TX FIFO is started, if there is no valid data in TX FIFO, the tx underrun error will be triggered. Therefore, the software configuration sequence must be followed.

15.4.4 Audio Channel Selector

Audio PWM only supports the playback of one channel of audio data. To play stereo audio, you need to use the audio channel selector module. The typical application of this module is to transfer the audio files from memory to TX FIFO through DMA to play stereo audio using Audio PWM. At this time, the user only needs to transfer the stereo data to TX FIFO in turn using DMA, and can select Play Left Channel, Play Right Channel, Sum of Two Channels' Data or Average of Two Channels' Data through the register dac_mix_sel in awpwm_1. This function is used together with the tx_ch_en in awpwm_fifo_ctrl, as shown below.

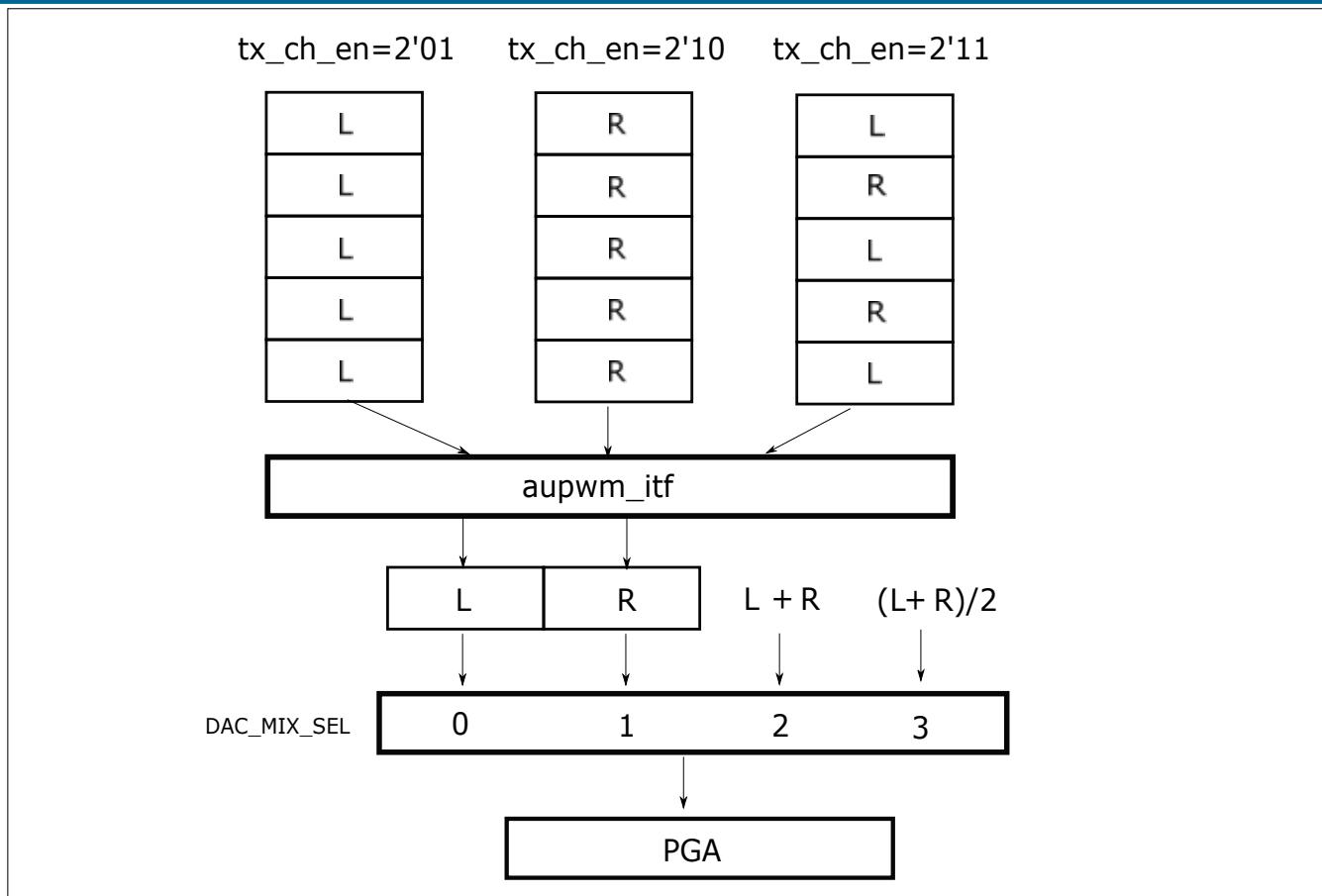


Fig. 15.3: Mixer

The above figure shows how to fill the data transferred using DMA in turn in the L and R caches through tx_ch_en. When the playback source is mono, you can select tx_ch_en=2' 01 or tx_ch_en=2' 10, and the data will be filled in R/L in turn. The data will not be updated to the other L/R. At this time, you can select the corresponding channel through a mixer selector to go to the next-stage modulation circuit.

When the playback source is stereo, select tx_ch_en=2' 11. At this time, the data transferred using DMA will be filled in L and R caches in turn, ensuring that both L and R will be updated. The storage format of stereo audio data source is also in the form of “L, R, L, R” arranged in turn, so the corresponding audio data will be correctly updated to L and R. At this time, you can choose which channel you need through a mixer selector, or sum or average the two channels’ data to go to the next-stage modulation circuit.

Pay attention to the selection of tx_ch_en and mixer. If the mono data is filled in the left channel through tc_ch_en, but you select the right channel for the mixer, null data will be played, resulting in an error.

15.4.5 Volume Control

The user can configure the volume through the register `dac_s0_volume` in `awpwm_s0`. The gain range is 95.5dB~18dB, and the register value*0.5 is the true gain value. Writing “1” in the register `dac_s0_volume_update` to update the operation after configuration.

Users can select the volume adjustment mode through `dac_s0_mute_softmode` or `dac_s0_ctrl_mode`. 0 indicates immediate adjustment without gradient processing; 1 means gradient adjustment in cross zero manner; 2 means gradient adjustment in a ramp manner. The registers `dac_s0_mute_rmpup_rate` and `dac_s0_ctrl_zcd_rate` adjust the slope of ramp or cross zero.

15.4.6 ZeroDetect

AudioPWM provides the ZeroDetect function. When this function is enabled, namely `aupwm_zd_0[16]=1`, the data in TX FIFO is always 0 and this closes the digital channel, with output being “0”. The purpose is to reduce the noise caused by playing 0 data and make the audio system more stable.

15.5 Configuration Process

1. Depending on the sampling rate of the audio to be played, select the corresponding sampling rate through `aupwm_0[31:28]`.
2. Depending on the number of audio channels for playback, decide the configuration of the mixer register.
3. Configure the DMA to transfer data to AudioPWM TX FIFO.
4. Enable the state machine by the `tx_ch_en` in `aupwm_fifo_ctrl`, and start playing.
5. Adjust the volume during playback (optional)

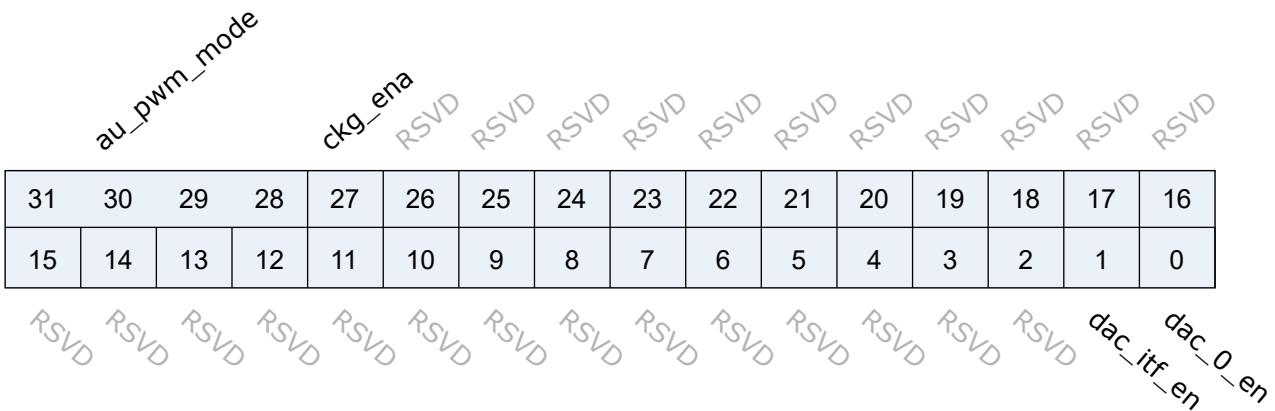
15.6 Register description

Name	Description
<code>aupwm_0</code>	
<code>aupwm_status</code>	
<code>aupwm_s0</code>	
<code>aupwm_s0_misc</code>	
<code>aupwm_zd_0</code>	
<code>aupwm_1</code>	
<code>aupwm_rsvd</code>	
<code>aupwm_test_0</code>	

Name	Description
aupwm_test_1	
aupwm_test_2	
aupwm_test_3	
aupwm_fifo_ctrl	
aupwm_fifo_status	
aupwm_fifo_data	

15.6.1 aupwm_0

Address: 0x20055000



Bits	Name	Type	Reset	Description
31:28	au_pwm_mode	r/w	4'd0	0:pwm 8k, 1:pwm 16k, 2:pwm 32k, 3:pwm 24k, 4:pwm 48k, 5:pwm 22.05k, 6:pwm 44.1k, 7:pwm 44.1k, 8:gpdac 8k, 9:gpdac 16k, 10:gpdac 32k, 11:gpdac 24k, 12:gpdac 48k, 13:gpdac 22.05k, 14:gpdac 44.1k, 15:gpdac 44.1k
27	ckg_ena	r/w	1	1:clock gen enable
26:2	RSVD			
1	dac_itf_en	r/w	0	1:enable dac to audio dma interface
0	dac_0_en	r/w	0	1:enable dac ch0

15.6.2 aupwm_status

Address: 0x20055004

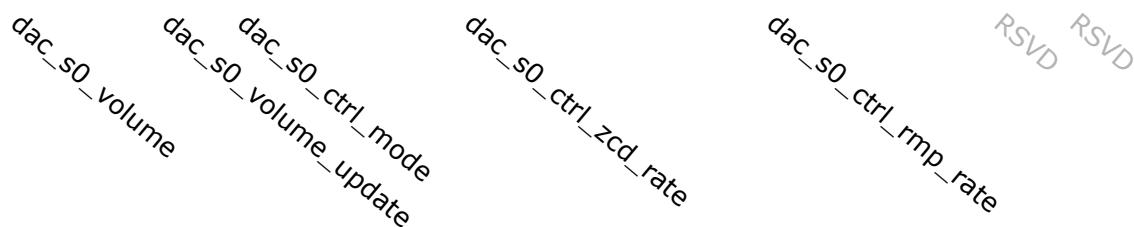
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	dac_s0_int_clr	dac_s0_int	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
RSVD	RSVD	dac_h0_busy	RSVD																
		dac_h0_mute_done																	

Bits	Name	Type	Reset	Description
31:25	RSVD			
24	audio_int_all	r	0	1:mute signal to analog
23	zd_amute	r	0	1:mute signal to analog
22:18	RSVD			
17	dac_s0_int_clr	r/w	0	1: clear interrupt
16	dac_s0_int	r	0	interrupt value
15:14	RSVD			
13	dac_h0_mute_done	r	1	1:dvga ch0 mute done
12	dac_h0_busy	r	0	1:dvga ch0 busy
11:0	RSVD			

15.6.3 aupwm_s0

Address: 0x20055008

dac_s0_volume															
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



Bits	Name	Type	Reset	Description
31	dac_s0_mute	r/w	0	dac dpga ch0 sw volume control 1:mute
30	dac_s0_mute_softmode	r/w	1	0:mute directly, 1:mute with ramp down
29:26	dac_s0_mute_rmpdn_rate	r/w	4'd6	mute ramp down rate: 0:2 fs sample 1:4 fs sample 2:8 fs sample 3:16 fs sample 4:32 fs sample 5:64 fs sample 6:128 fs sample 7:256 fs sample 8:512 fs sample 9:1024 fs sample 8:2048 fs sample

Bits	Name	Type	Reset	Description
25:22	dac_s0_mute_rmpup_rate	r/w	4'd0	mute ramp up rate 0:2 fs sample 1:4 fs sample 2:8 fs sample 3:16 fs sample 4:32 fs sample 5:64 fs sample 6:128 fs sample 7:256 fs sample 8:512 fs sample 9:1024 fs sample 8:2048 fs sample
21:13	dac_s0_volume	r/w	9'd0	volume s9.1, -95.5dB +18dB in 0.5dB step
12	dac_s0_volume_update	r/w	0	1:volume update
11:10	dac_s0_ctrl_mode	r/w	2'd2	0:direct force volume, 1:update volume at zero crossing, 2:update volume with ramp
9:6	dac_s0_ctrl_zcd_rate	r/w	4'd2	zero crossing rate 0:2 fs sample 1:4 fs sample 2:8 fs sample 3:16 fs sample 4:32 fs sample 5:64 fs sample 6:128 fs sample 7:256 fs sample 8:512 fs sample 9:1024 fs sample 8:2048 fs sample
5:2	dac_s0_ctrl_rmp_rate	r/w	4'd6	ramp rate 0:2 fs sample 1:4 fs sample 2:8 fs sample 3:16 fs sample 4:32 fs sample 5:64 fs sample 6:128 fs sample 7:256 fs sample 8:512 fs sample 9:1024 fs sample 8:2048 fs sample

Bits	Name	Type	Reset	Description
1:0	RSVD			

15.6.4 aupwm_s0_misc

Address: 0x20005500c

dac_s0_ctrl_zcd_timeout															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	RSVD														
14	RSVD														
13	RSVD														
12	RSVD														
11	RSVD														
10	RSVD														
9	RSVD														
8	RSVD														
7	RSVD														
6	RSVD														
5	RSVD														
4	RSVD														
3	RSVD														
2	RSVD														
1	RSVD														
0	RSVD														

Bits	Name	Type	Reset	Description
31:28	dac_s0_ctrl_zcd_timeout	r/w	4'd4	zero crossing time out period 0:2 fs sample 1:4 fs sample 2:8 fs sample 3:16 fs sample 4:32 fs sample 5:64 fs sample 6:128 fs sample 7:256 fs sample 8:512 fs sample 9:1024 fs sample 8:2048 fs sample
27:0	RSVD			

15.6.5 aupwm_zd_0

Address: 0x20055010

RSVD	zd_en																
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		

zd_time

RSVD

Bits	Name	Type	Reset	Description
31:17	RSVD			
16	zd_en	r/w	0	two channels, 0: zd disabled, 1: zd on channel0, 2: zd on channel1, 3: zd on both channels
15	RSVD			
14:0	zd_time	r/w	15'd512	number of zeros

15.6.6 aupwm_1

Address: 0x20055014

RSVD	dac_dsm_dither_prbs_mode																
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		

dac_dsm_dither_en dac_dsm_dither_amp RSVD dac_dsm_scaling_en dac_dsm_scaling_mode dac_dsm_order RSVD dac_dsm_out_fmt RSVD dac_mix_sel
 dac_dsm_dither_prbs_mode dac_dsm_dither_en dac_dsm_dither_amp RSVD dac_dsm_scaling_en dac_dsm_scaling_mode dac_dsm_order RSVD dac_dsm_out_fmt RSVD dac_mix_sel

Bits	Name	Type	Reset	Description
31:17	RSVD			
16:15	dac_dsm_dither_prbs_mode	r/w	0	dac dsm dither lfsr mode:0:LFSR32, 1:LFSR24, 2:LFSR16, 3:LFSR12
14	dac_dsm_dither_en	r/w	1	1:enable dac dsm dither
13:11	dac_dsm_dither_amp	r/w	0	dac dsm dither amplitue
10	dac_dsm_scaling_en	r/w	1	1:enable dac dsm scaling
9	RSVD			
8:7	dac_dsm_scaling_mode	r/w	0	dac dsm scaling value; u4.4
6:5	dac_dsm_order	r/w	0	0: 2-order, 1: 3-order
4	dac_dsm_out_fmt	r/w	0	0:offset binary 1:2's complement
3:2	RSVD			
1:0	dac_mix_sel	r/w	0	0: L channel, 1:R channel, 2: L+R, 3: (L+R)/2

15.6.7 aupwm_rsvid

Address: 0x20055018

au_pwm_reserved

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

au_pwm_reserved

Bits	Name	Type	Reset	Description
31:0	au_pwm_reserved	r/w	32'hffff0000	

15.6.8 aupwm_test_0

Address: 0x2005501c

dac_dpga_0

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

dac_in_0

Bits	Name	Type	Reset	Description
31:16	dac_dpga_0	r	0	
15:0	dac_in_0	r	0	

15.6.9 aupwm_test_1

Address: 0x200055020

RSVD	dac_fir_0																		
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				dac_fir_0

Bits	Name	Type	Reset	Description
31:17	RSVD			
16:0	dac_fir_0	r	0	

15.6.10 aupwm_test_2

Address: 0x200055024

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------------|
| 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 | | | | dac_sinc_0 |

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	dac_sinc_0	r	0	

15.6.11 aupwm_test_3

Address: 0x20055028

au_pwm_test_read

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

au_pwm_test_read

Bits	Name	Type	Reset	Description
31:0	au_pwm_test_read	r	0	

15.6.12 aupwm_fifo_ctrl

Address: 0x2005508c

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	tx_data_mode	RSVD	RSVD	RSVD	tx_trg_level					
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

tx_dra_cnt *RSVD* *RSVD* *RSVD* *RSVD* *tx_ch_en* *RSVD* *RSVD* *RSVD* *tx_dra_en* *txa_int_en* *txu_int_en* *txo_int_en* *tx_fifo_flush*

Bits	Name	Type	Reset	Description
31:26	RSVD			
25:24	tx_data_mode	r/w	2'b0	TX_FIFO_DATIN_MODE. TX FIFO DATA Input Mode (Mode 0, 1, 2, 3) Mode 0: Valid data's MSB is at [31] of TX_FIFO register Mode 1: Valid data's MSB is at [23] of TX_FIFO register Mode 2: Valid data's MSB is at [19] of TX_FIFO register Mode 3: Valid data's MSB is at [15] of TX_FIFO register For 16-bits transmitted audio sample: Mode 0: FIFO_I[15:0] = TXDATA[31:16] Mode 1: FIFO_I[15:0] = TXDATA[23:8] Mode 2: FIFO_I[15:0] = TXDATA[19:4] Mode 3: FIFO_I[15:0] = TXDATA[15:0]

Bits	Name	Type	Reset	Description
23:21	RSVD			
20:16	tx_trg_level	r/w	5'd7	<p>TX_FIFO_TRG_LEVEL.</p> <p>TX FIFO Trigger Level (TXTL[4:0])</p> <p>Interrupt and DMA request trigger level for TX FIFO room available condition</p> <p>IRQ/DRQ Generated when WLEVEL > TXTL[4:0]</p> <p>Notes:</p> <p>WLEVEL represents the number of room available in the TX FIFO</p>
15:14	tx_drq_cnt	r/w	2'b0	<p>DAC_DRQ_CLR_CNT.</p> <p>When TX FIFO available room less than or equal N, DRQ Request will be de-asserted. N is defined here:</p> <ul style="list-style-type: none"> 00: IRQ/DRQ de-asserted when WLEVEL <= TXTL[4:0] 01: IRQ/DRQ de-asserted when WLEVEL < 2 10: IRQ/DRQ de-asserted when WLEVEL < 4 11: IRQ/DRQ de-asserted when WLEVEL < 8 <p>WLEVEL represents the number of room available in the TX FIFO</p>
13:10	RSVD			
9:8	tx_ch_en	r/w	2'b0	<p>TX_FIFO_DATOUT_DST.</p> <p>TX FIFO Data Output Destination Select.</p> <p>0: Disable 1: Enable</p> <p>Bit9: DAC2 data</p> <p>Bit8: DAC1 data</p> <p>When some of the above bits set to ' 1' , these data are always arranged in order from low-bit to high-bit.(bit8->bit9)</p>
7:5	RSVD			
4	tx_drq_en	r/w	1'b0	<p>DAC_DRQ_EN.</p> <p>DAC FIFO Room Available DRQ Enable.</p> <p>0: Disable</p> <p>1: Enable</p>
3	txa_int_en	r/w	1'b0	<p>DAC_IRQ_EN.</p> <p>DAC FIFO Room Available IRQ Enable.</p> <p>0: Disable</p> <p>1: Enable</p>
2	txu_int_en	r/w	1'b0	<p>DAC_UNDERRUN_IRQ_EN.</p> <p>DAC FIFO Under Run IRQ Enable</p> <p>0: Disable</p> <p>1: Enable</p>

Bits	Name	Type	Reset	Description
1	txo_int_en	r/w	1'b0	DAC_OVERRUN_IRQ_EN. DAC FIFO Over Run IRQ Enable 0: Disable 1: Enable
0	tx_fifo_flush	w1c	1'b0	DAC_FIFO_FLUSH. DAC FIFO Flush. Write ‘1’ to flush TX FIFO, self clear to ‘0’ .

15.6.13 aupwm_fifo_status

Address: 0x20055090

RSVD	txa	RSVD	RSVD	RSVD	txa_cnt						
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

RSVD	txa_int	RSVD	txu_int	txo_int	RSVD								
------	------	------	------	------	------	------	------	------	---------	------	---------	---------	------

Bits	Name	Type	Reset	Description
31:25	RSVD			
24	txa	r	1'b1	TXA. TX FIFO Room Available 0: No room for new sample in TX FIFO 1: More than one room for new sample in TX FIFO (>= 1 word)
23:21	RSVD			
20:16	txa_cnt	r	5'd16	TXA_CNT. TX FIFO Available Room Word Counter
15:5	RSVD			
4	txa_int	r	1'b0	TXA_INT. TX FIFO Room Available Pending Interrupt 0: No Pending IRQ 1: Room Available Pending IRQ Automatic clear if interrupt condition fails.
3	RSVD			

Bits	Name	Type	Reset	Description
2	txu_int	r	1'b0	TXU_INT. TX FIFO Underrun Pending Interrupt 0: No Pending IRQ 1: FIFO Underrun Pending IRQ Write ‘1’ to clear this interrupt
1	txo_int	r	1'b0	TXO_INT. TX FIFO Overrun Pending Interrupt 0: No Pending IRQ 1: FIFO Overrun Pending IRQ Write ‘1’ to clear this interrupt
0	RSVD			

15.6.14 aupwm_fifo_data

Address: 0x20055094

tx_data

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

tx_data

Bits	Name	Type	Reset	Description
31:0	tx_data	w	32'h0	TX_DATA. Transmitting left, right channel sample data should be written this register one by one. The left channel sample data is first and then the right channel sample.

16.1 Overview

A 16-bit ADC is built in the chip to sample the digital PDM signals or analog signals of the audio.

16.2 Features

- One 16-bit ADC is integrated to support 1 analog mic differential/single-ended input and multiplexed GPIO input.
 - * Sampling rate: 8k~96k
 - * Signal to noise ratio (AW): 90 dB @ 6 dB gain
 - * Harmonic distortion + noise: 80dB @ 6dB gain
 - * Analog pre-amplifier gain: 6~42 dB, 3 dB per gear
- Adjustable high-pass filter and independent digital volume control
- Supports digital mic interface, with GPIO input multiplexed
- Supports DC measurement mode with accuracy of 16-bit
- Independent digital volume control
- 32-bit FIFO with a depth of 32
- Supports DMA transfer mode

16.3 Clock Tree

The user starts the Audio PLL to select the corresponding frequency value. After one frequency division, the user enters the module and selects the division factor through the `adc_rate` in `audpdm_top`. The value of this register and the frequency division ratio are shown as follows.

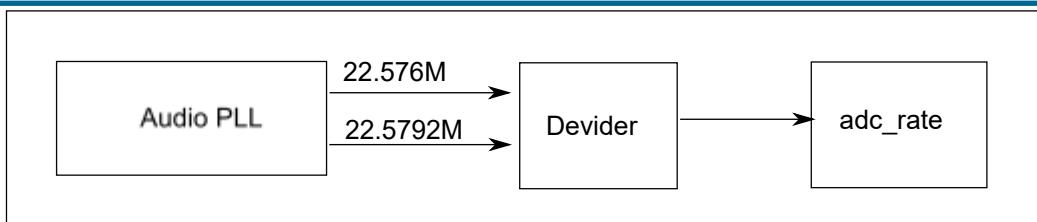


Fig. 16.1: Block Diagram of Clock

16.4 Functional Description

The block diagram of AudioADC is shown as follows.

BL616 ADUO & ADDA

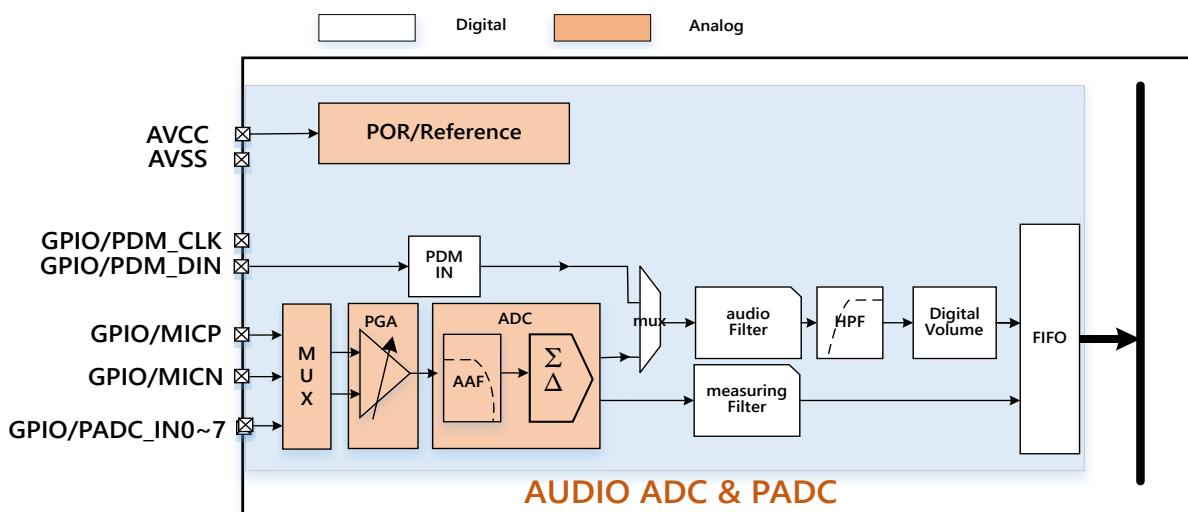


Fig. 16.2: Block Diagram of Module

The input signal of AUADC can be either a digital PDM signal or an analog signal (determined by pdm_dac_0[12]). When a PDM digital signal is selected, the input enters PDM demodulator and then passes through a filter, and the volume controller enters RX FIFO. When a digital signal is selected, it passes through a PGA, an ADC converter and the corresponding filter and finally enter RX FIFO.

16.4.1 Selection of PDM Left and Right Channels

When the input signal is in the digital PDM format, you can choose to record the left or right channel of PDM through the adc_0_pdm_sel in PDM_PDM_0.

16.4.2 AUADC Interrupt

AUADC supports the following interrupt control modes:

- RX FIFO request interrupt
- RX FIFO underrun interrupt
- RX FIFO overrun interrupt

A RX FIFO request interrupt is generated when RX_DRQ_CNT in RX_FIFO_CTRL is greater than RX_TRG_LEVEL. When the condition is not met, the interrupt flag is cleared automatically.

When there is no data in RX FIFO, but the user enables RX FIFO modulation through RX_CH_EN in RX_FIFO_CTRL, the RX FIFO underrun interrupt is generated.

When the user fills in data that exceeds the maximum depth of RX FIFO, it leads to RX FIFO overflow and cause a RX FIFO overrun interrupt.

16.4.3 FIFO Format Control

The register AUADC_RX_FIFO_CTRL can control the format of the audio data to be stored in FIFO.

The FIFO controller supports the following four data storage formats, which are determined by FIFO_CTRL[25:24].

- Mode 0:

DATA[31:0] = {FIFO[15:0],16' h0}

- Mode 1:

DATA[31:0] = {8{FIFO[15]},FIFO[15:0],8' h0}

- Mode 2:

DATA[31:0] = {12{FIFO[15]},FIFO[15:0],4' h0}

- Mode 3:

DATA[31:0] = {16{FIFO[15]},FIFO[15:0]}

Distribution of MSB

- Mode 0:

The MSB of data is 31 bits

- Mode 1:

The MSB of data is 23 bits

- Mode 2:

The MSB of data is 19 bits

- Mode 3:

The MSB of data is 15 bits

If there is no special requirement for the storage format, generally, Mode3 is appropriate. As the maximum resolution of ADC is 16-bit, using 16-bit RAM to store audio can achieve the greatest efficiency. For other formats, the valid 16-bit data is placed in different positions in the 32-bit width, with low bits filled with 0 and high bits filled with sign bits.

16.4.4 Startup of FIFO and DMA Transfer

The data in FIFO of the PDM module can be transferred by DMA.

The user can obtain the current amount of valid data in FIFO in real time through the register PDM_RX_FIFO_STATUS.

The FIFO count threshold (8/16/32) for initiating DMA request is selected by configuring FIFO_CTRL[15:14], or can be determined by FIFO_CTRL[22:16].

When the count value is greater than the set threshold, and the FIFO of the channel corresponding to PDM_RX_FIFO_CTRL[12:8] is enabled, a DMA transfer is initiated.

When TX FIFO is started, if there is no valid data in TX FIFO, the tx underrun error will be triggered. Therefore, the software configuration sequence must be followed.

16.5 Configuration Process

1. For the sampling rate of the recorded audio, select the corresponding sampling rate through audpdm_top[31:28].
2. Configure the adc_0_src register of pdm_dac_0 depending on whether the recorded data source is PDM digital signal or analog signal.
3. In case of pdm format, select the channel of pdm through the adc_0_pdm_sel in pdm_pdm_0
4. Configure the DMA to transfer the RX FIFO data of Audio to the designated area in real time
5. Turn on the state machine through the rx_ch_en in audadc_rx_fifo_ctrl to start recording
6. Adjust the volume during recording (optional)

16.6 Register description

Name	Description
audpdm_top	
audpdm_itf	
pdm_adc_0	
pdm_adc_1	
pdm_dac_0	
pdm_pdm_0	
pdm_dbg_0	
pdm_dbg_1	
pdm_dbg_2	
pdm_adc_s0	
audadc_ana_cfg1	
audadc_ana_cfg2	
audadc_cmd	
audadc_data	
audadc_rx_fifo_ctrl	
audadc_rx_fifo_status	
audadc_rx_fifo_data	

16.6.1 audpdm_top

Address: 0x2000ac00

adc_rate															
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
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	pdm_itf_inv_sel	adc_itf_inv_sel	RSVD	audio_ckg_en

Bits	Name	Type	Reset	Description
31:28	adc_rate	r/w	4'd1	adc fs 0:8kHz, 1:16kHz, 2:24kHz(22.05k), 3:32kHz, 4:48kHz(44.1k), 5:96kHz, 6:reserved, 7:manual, 8:padc 128kHz, 9:padc 256kHz, 10:padc 512kHz
27:4	RSVD			
3	pdm_itf_inv_sel	r/w	1'd0	1:invert clk_pdm_inf
2	adc_itf_inv_sel	r/w	1'd0	1:invert clk_adc_itf
1	RSVD			
0	audio_ckg_en	r/w	1'd0	1:enable audio clock generator

16.6.2 audpdm_itf

Address: 0x2000ac04

adc_itf_en															
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
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

Bits	Name	Type	Reset	Description
31	RSVD			
30	adc_itf_en	r/w	1'd0	1:enable adc to audio dma interface
29:1	RSVD			

Bits	Name	Type	Reset	Description
0	adc_0_en	r/w	1'd0	1:enable adc ch0

16.6.3 pdm_adc_0

Address: 0x2000ac08

Bits	Name	Type	Reset	Description
31:1	RSVD			
0	adc_0_fir_mode	r/w	1'd0	adc fir mode

16.6.4 pdm_adc_1

Address: 0x2000ac0c

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | | | |

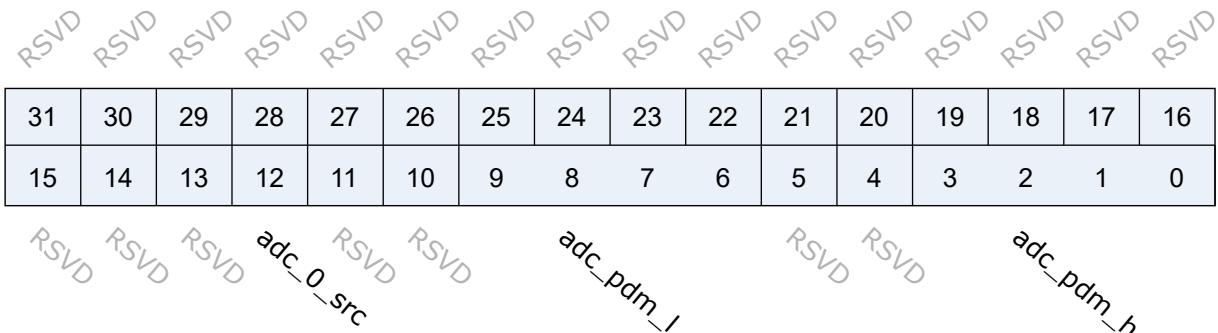
Bits	Name	Type	Reset	Description
31:10	RSVD			
9	adc_0_k2_en	r/w	1'd0	adc ch0 hpf parameter k2 enable
8:5	adc_0_k2	r/w	4'd13	adc ch0 hpf parameter k2



Bits	Name	Type	Reset	Description
4	RSVD			
3:0	adc_0_k1	r/w	4'd8	adc ch0 hpf parameter k1

16.6.5 pdm_dac_0

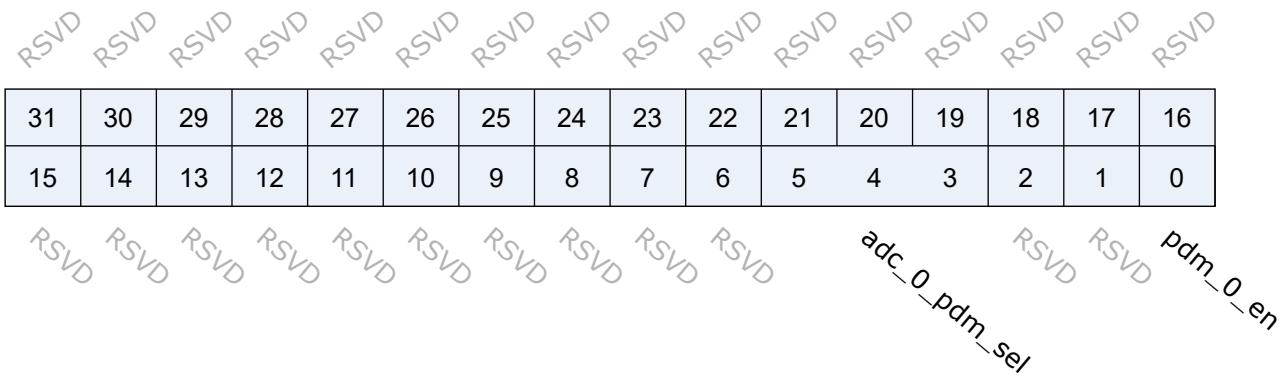
Address: 0x2000ac10



Bits	Name	Type	Reset	Description
31:13	RSVD			
12	adc_0_src	r/w	1'd0	0:adc, 1:pdm
11:10	RSVD			
9:6	adc_pdm_l	r/w	4'b1010	pdm low value
5:4	RSVD			
3:0	adc_pdm_h	r/w	4'b0110	pdm high value

16.6.6 pdm_pdm_0

Address: 0x2000ac1c



Bits	Name	Type	Reset	Description
31:6	RSVD			
5:3	adc_0_pdm_sel	r/w	3'd0	adc ch0 source select: 0:pdm_0_l, 1:pdm_0_r, 2:pdm_1_l, 3:pdm_1_r, 4:pdm_2_l, 5:pdm_2_r
2:1	RSVD			
0	pdm_0_en	r/w	1'd0	1:enable pdm_0

16.6.7 pdm_dbg_0

Address: 0x2000ac24

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

aud_test_read_sel adc_test_din_en adc_test_ckin_en

RSVD RSVD

Bits	Name	Type	Reset	Description
31:30	RSVD			
29:24	aud_test_read_sel	r/w	6'd0	select aud_test_read(0x28) test point
23	adc_test_din_en	r/w	1'd0	1:enable adc test data input from GPIO
22	RSVD			
21	adc_test_ckin_en	r/w	1'd0	1:enable adc test clkok input from GPIO
20:0	RSVD			

16.6.8 pdm_dbg_1

Address: 0x2000ac28

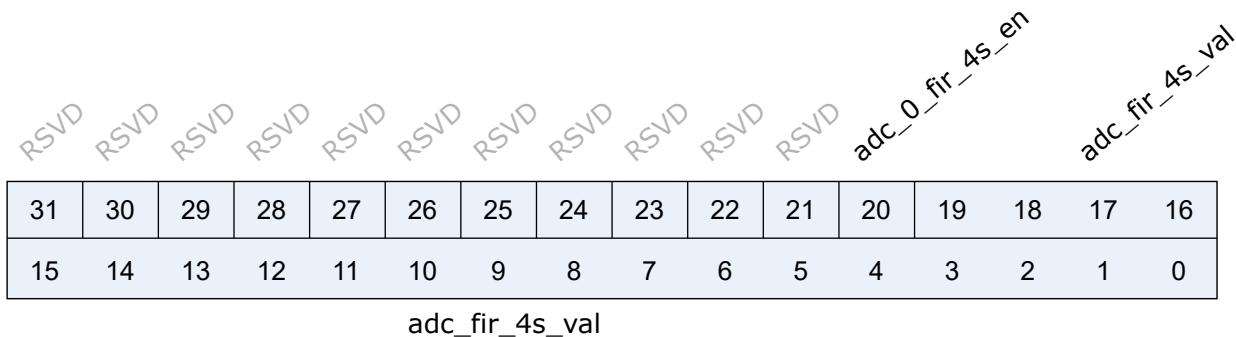
aud_test_read															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

aud_test_read

Bits	Name	Type	Reset	Description
31:0	aud_test_read	r	32'd0	audio test read value

16.6.9 pdm_dbg_2

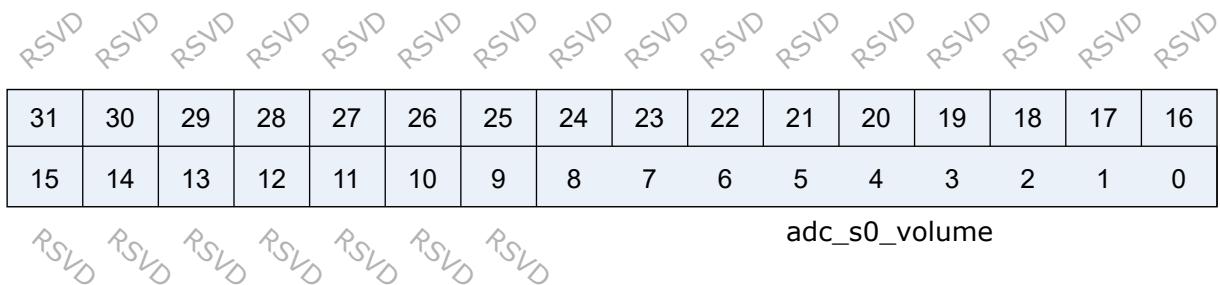
Address: 0x2000ac2c



Bits	Name	Type	Reset	Description
31:21	RSVD			
20	adc_0_fir_4s_en	r/w	1'd0	1:force adc ch0 fir output
19:0	adc_fir_4s_val	r/w	20'd0	force value of adc fir output

16.6.10 pdm_adc_s0

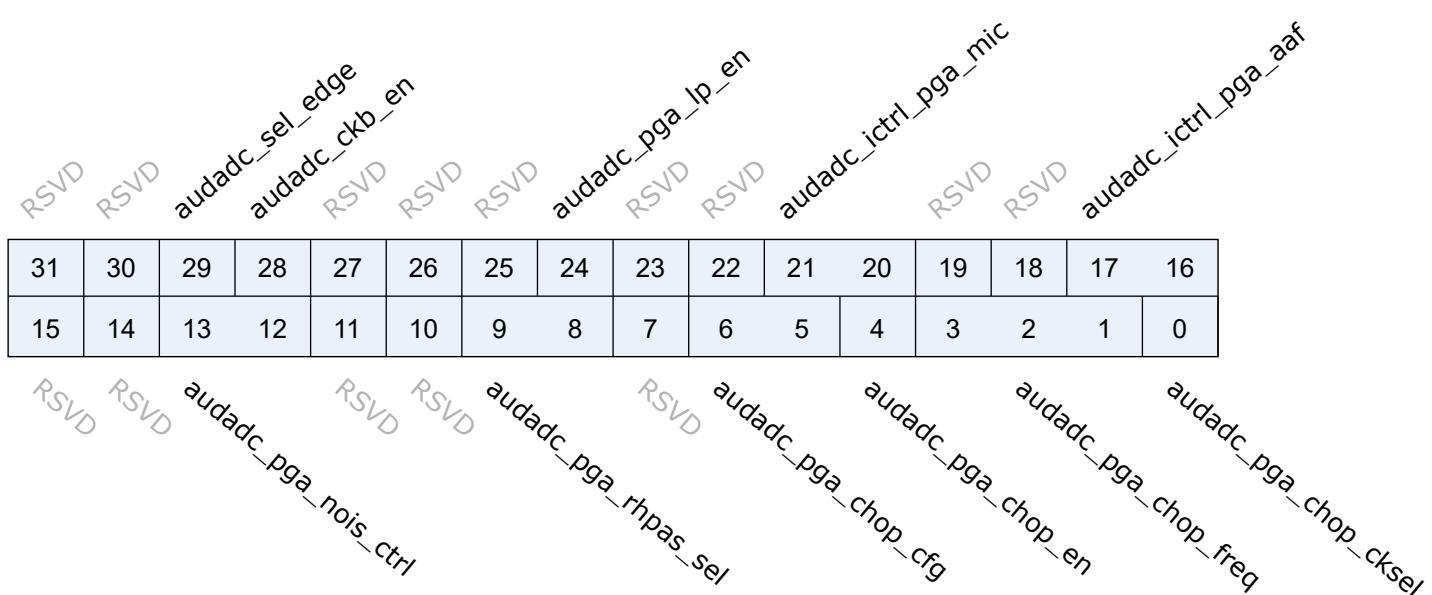
Address: 0x2000ac38



Bits	Name	Type	Reset	Description
31:9	RSVD			
8:0	adc_s0_volume	r/w	9'd0	volume s9.1, -95.5dB +18dB in 0.5dB step

16.6.11 audadc_ana_cfg1

Address: 0x2000ac60

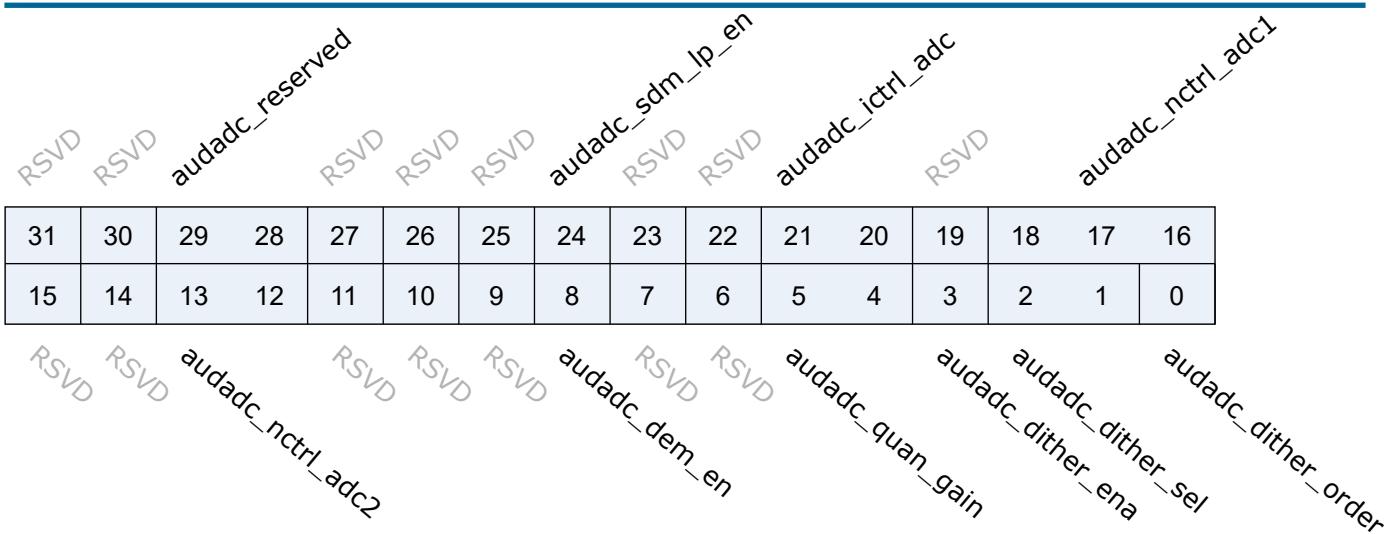


Bits	Name	Type	Reset	Description
31:30	RSVD			
29	audadc_sel_edge	r/w	1'h0	ADC output data clock edge 0 = falling edge sent, rising edge receive 1 = rising edge sent, falling edge receive
28	audadc_ckb_en	r/w	1'h0	AUDADC clock phase control 0 = 0° 1 = 180°
27:25	RSVD			
24	audadc_pga_lp_en	r/w	1'h0	PGA lowpower funciton reversed, not realized in circuit
23:22	RSVD			
21:20	audadc_ictrl_pga_mic	r/w	2'h1	PGA_OPMIC bias current control 00 = 4uA 01 = 5uA 10 = 6uA 11 = 7uA
19:18	RSVD			
17:16	audadc_ictrl_pga_aaf	r/w	2'h1	PGA_OPAAF bias current control 00 = 4uA 01 = 5uA 10 = 6uA 11 = 7uA
15:14	RSVD			
13:12	audadc_pga_nois_ctrl	r/w	2'h0	PGA noise control when configured to single-ended not used

Bits	Name	Type	Reset	Description
11:10	RSVD			
9:8	audadc_pga_rhpas_sel	r/w	2'h0	PGA high pass filter R control when configured to AC-coupled mode 00 = 480kΩ 01 = 320kΩ 10 = 160kΩ 11 = 4kΩ, fast startup
7	RSVD			
6:5	audadc_pga_chop_cfg	r/w	2'h3	control chopper for opmic&opaaf 00 = opmic off & opaaf off 01 = opmic off & opaaf on 10 = opmic on & opaaf off 11 = opmic on & opaaf on
4	audadc_pga_chop_en	r/w	1'h1	PGA chopper control 0 = disable 1 = enable
3:1	audadc_pga_chop_freq	r/w	3'h4	PGA chopper frequency control @Fs=2048k 000 = 8k 001 = 16k 010 = 32k 011 = 64k 100 = 128k 101 = 256k 110 = 512k 111 = 1024k
0	audadc_pga_chop_cksel	r/w	1'h0	PGA chopper clock source selection 0 = adc clock 1 = synchronized clock from SDM not used

16.6.12 audadc_ana_cfg2

Address: 0x2000ac64

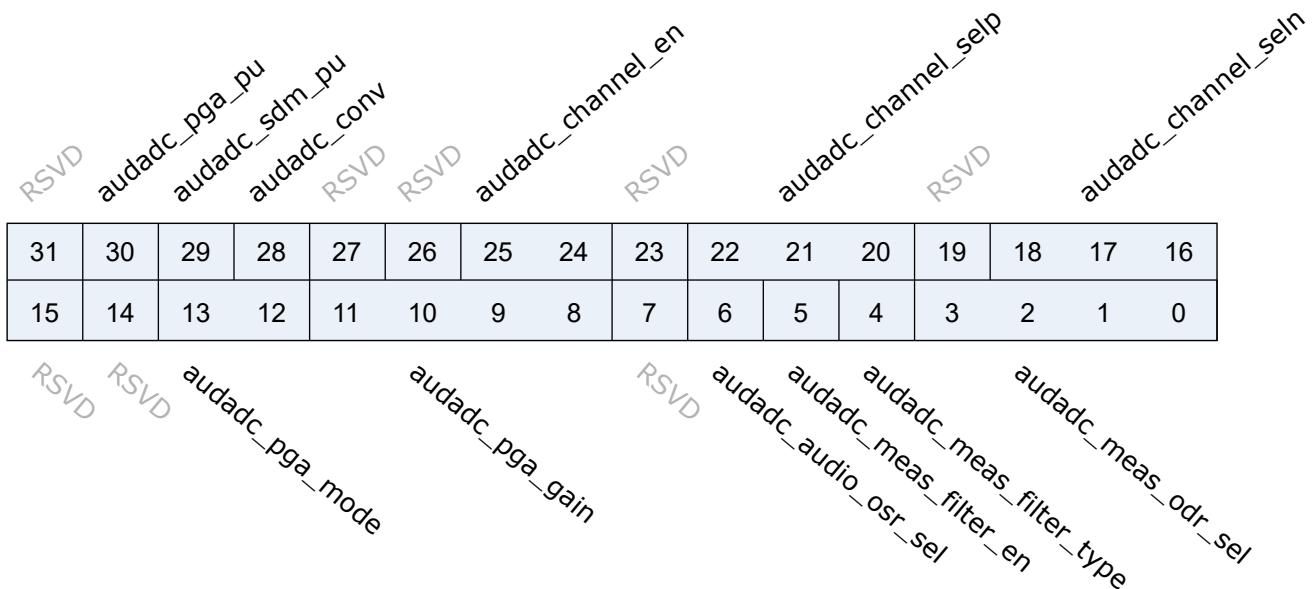


Bits	Name	Type	Reset	Description
31:30	RSVD			
29:28	audadc_reserved	r/w	2'h0	AUDADC reserved register
27:25	RSVD			
24	audadc_sdm_lp_en	r/w	1'h0	SDM lowpower funciton 0 = disable 1 = enable, 0.6 of disable
23:22	RSVD			
21:20	audadc_ictrl_adc	r/w	2'h1	SDM bias current control 00 = 4uA 01 = 5uA 10 = 6uA 11 = 7uA
19	RSVD			
18:16	audadc_nctrl_adc1	r/w	3'h3	op number control for first integrator in SDM 000 = 1(12uA) 001 = 2(24uA) 010 = 3(36uA) 011 = 4(48uA) 100 = 5(60uA) 101 = 5(60uA) 110 = 5(60uA) 111 = 5(60uA)
15:14	RSVD			
13:12	audadc_nctrl_adc2	r/w	2'h1	op number control for second integrator in SDM 00 = 1(12uA) 01 = 2(24uA) 10 = 3(36uA) 11 = 3(36uA)
11:9	RSVD			
8	audadc_dem_en	r/w	1'h1	dem function control 0 = disable 1 = enable
7:6	RSVD			

Bits	Name	Type	Reset	Description
5:4	audadc_quan_gain	r/w	2'h1	quantizer gain control for SDM 00 = Vref/14 01 = Vref/12 10 = Vref/10 11 = Vref/8
3	audadc_dither_ena	r/w	1'h1	dither control 0 = disable 1 = enable
2:1	audadc_dither_sel	r/w	2'h2	dither level control for SDM 00 = 0 01 = LSB*1/15 10 = LSB*2/15 11 = LSB*3/15
0	audadc_dither_order	r/w	1'h0	dither order control for SDM 0 = 0 order 1 = 1 order

16.6.13 audadc_cmd

Address: 0x2000ac68



Bits	Name	Type	Reset	Description
31	RSVD			
30	audadc_pga_pu	r/w	1'h0	PGA related circuit enable 0 = disable 1 = enable
29	audadc_sdm_pu	r/w	1'h0	SDM related circuit enable 0 = disable 1 = enable

Bits	Name	Type	Reset	Description
28	audadc_conv	r/w	1'h0	<p>SDM conversion start singal</p> <p>0 = remain resetting status 1 = start conversion</p> <p>both analog intetrator and measuring digital decimation filter will be reset when</p> <p>audadc_conv configured to low. Measuring digital decimation filter need to reset</p> <p>to same initial condition because it's feedback configuration for FIR. Audio filter</p> <p>dont need this resetting.</p>
27:26	RSVD			
25:24	audadc_channel_en	r/w	2'h0	<p>channel mux switch enable or disable</p> <p>MSB controls Positive channel, LSB controls Negative channel</p> <p>0 = disable, look into each channel will see high impedance</p> <p>1 = enable, one of eight channel will be choose</p>
23	RSVD			
22:20	audadc_channel_selp	r/w	3'h0	<p>Positive channel selection, connected to PGA positive terminal</p> <p>000 = AIN0 001 = AIN1</p> <p>010 = AIN2 011 = AIN3</p> <p>100 = AIN4 101 = AIN5</p> <p>110 = AIN6 111 = AIN7</p>
19	RSVD			
18:16	audadc_channel_seln	r/w	3'h0	<p>Negative channel selection, connected to PGA negative terminal</p> <p>000 = AIN0 001 = AIN1</p> <p>010 = AIN2 011 = AIN3</p> <p>100 = AIN4 101 = AIN5</p> <p>110 = AIN6 111 = AIN7</p>
15:14	RSVD			
13:12	audadc_pga_mode	r/w	2'h0	<p>PGA mode configuration</p> <p>00: AC-Coupled & differential-ended, Audio application</p> <p>01: AC-Coupled & single-ended, Audio application</p> <p>10: DC-Coupled & differential-ended, Measuring application</p> <p>11: DC-Coupled & single-ended, Measuring application(may not used)</p>

Bits	Name	Type	Reset	Description
11:8	audadc_pga_gain	r/w	4'h0	PGA Gain control 0000 = 6dB 0001 = 6dB 0010 = 6dB 0011 = 9dB 0100 = 12dB 0101 = 15dB 0110 = 18dB 0111 = 21dB 1000 = 24dB 1001 = 27dB 1010 = 30dB 1011 = 33dB 1100 = 36dB 1101 = 39dB 1110 = 42dB 1111 = 42dB
7	RSVD			
6	audadc_audio_osr_sel	r/w	1'h0	audio osr configuration 0 = 128 1 = 64
5	audadc_meas_filter_en	r/w	1'h0	measuring mode enable, measuring filter is on when set to high 0 = disable 1 = enable
4	audadc_meas_filter_type	r/w	1'h0	digital dicimation filter selection when in measuring mode 0 = SINC3 1 = Low-Latency
3:0	audadc_meas_odr_sel	r/w	4'h3	audadc ouput data rate selection when configured to measuring mode 0000 = 2.5SPS 0001 = 5SPS 0010 = 10SPS 0011 = 20SPS 0100 = 25SPS 0101 = 50SPS 0110 = 100SPS 0111 = 200SPS 1000 = 400SPS 1001 = 800SPS 1010 = 1000SPS 1011 = 2000SPS 1100 = 4000SPS 1101 = 4000SPS 1110 = 4000SPS 1111 = 4000SPS

16.6.14 audadc_data

Address: 0x2000ac6c



audadc_valid_4s_en	audadc_valid_4s_val	audadc_soft_rst	RSVD	RSVD	RSVD	RSVD	audadc_data_rdy	audadc_raw_data
31	30	29	28	27	26	25	24	23
22	21	20	19	18	17	16	15	14

audadc raw data

Bits	Name	Type	Reset	Description
31	audadc_valid_4s_en	r/w	1'h0	
30	audadc_valid_4s_val	r/w	1'h0	
29	audadc_soft_rst	r/w	1'h0	
28:25	RSVD			
24	audadc_data_rdy	r	1'h0	audadc data ready indicator when measuring mode selected, auto reset to 0 after read 0 = not ready 1 = ready
23:0	audadc_raw_data	r	16'h0	audadc output 16bit data, 2's

16.6.15 audadc rx fifo ctrl

Address: 0x2000ac80

<i>RSVD</i>	<i>RSVD</i>	<i>RSVD</i>	<i>RSVD</i>	<i>RSVD</i>	<i>RSVD</i>	<i>rx_data_mode</i>	<i>RSVD</i>	<i>RSVD</i>	<i>RSVD</i>	<i>RSVD</i>	<i>rx_trg_level</i>				
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

Bits	Name	Type	Reset	Description
31:26	RSVD			

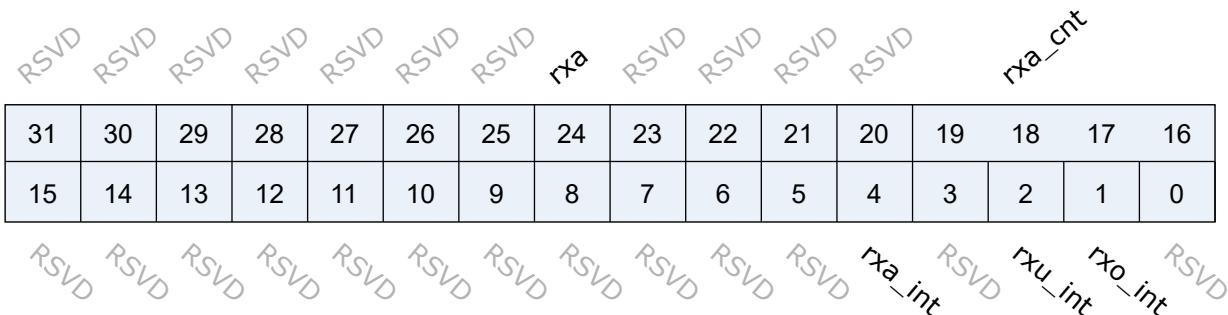
Bits	Name	Type	Reset	Description
25:24	rx_data_mode	r/w	2'b0	<p>RX_FIFO_DATOUT_MODE.</p> <p>RX FIFO DATA Output Mode (Mode 0, 1, 2, 3)</p> <p>Mode 0: Valid data's MSB is at [31] of RX_FIFO register</p> <p>Mode 1: Valid data's MSB is at [23] of RX_FIFO register</p> <p>Mode 2: Valid data's MSB is at [19] of RX_FIFO register</p> <p>Mode 3: Valid data's MSB is at [15] of RX_FIFO register</p> <p>Note: Expanding '0' at LSB of RX FIFO register (data invalid region)</p> <p>Expanding sign bit at MSB of RX FIFO register (data invalid region)</p> <p>For 24-bit received audio sample resolution:</p> <ul style="list-style-type: none"> Mode 0: RXDATA[31:0] = FIFO_O[23:0], 8' h0 Mode 1: RXDATA[31:0] = 8FIFO_O[23], FIFO_O[23:0] Mode 2: RXDATA[31:0] = 12FIFO_O[23], FIFO_O[23:4] Mode 3: RXDATA[31:0] = 16FIFO_O[23], FIFO_O[23:8] <p>For 20-bit received audio sample resolution:</p> <ul style="list-style-type: none"> Mode 0: RXDATA[31:0] = FIFO_O[23:4], 12' h0 Mode 1: RXDATA[31:0] = 8FIFO_O[23], FIFO_O[23:4], 4' h0 Mode 2: RXDATA[31:0] = 12FIFO_O[23], FIFO_O[23:4] Mode 3: RXDATA[31:0] = 16FIFO_O[23], FIFO_O[23:8] <p>For 16-bit received audio sample resolution:</p> <ul style="list-style-type: none"> Mode 0: RXDATA[31:0] = FIFO_O[23:8], 16' h0 Mode 1: RXDATA[31:0] = 8FIFO_O[23], FIFO_O[23:8], 8' h0 Mode 2: RXDATA[31:0] = 12FIFO_O[23], FIFO_O[23:8], 4'h0 Mode 3: RXDATA[31:0] = 16FIFO_O[23], FIFO_O[23:8]
23:20	RSVD			
19:16	rx_trg_level	r/w	4'd3	<p>RX_FIFO_TRG_LEVEL.</p> <p>RX FIFO Trigger Level (RXTL[3:0])</p> <p>Interrupt and DMA request trigger level for RX FIFO Data Available condition</p> <p>IRQ/DRQ Generated when WLEVEL > RXTL[3:0]</p> <p>Notes:</p> <p>WLEVEL represents the number of valid samples in the RX FIFO</p>

Bits	Name	Type	Reset	Description
15:14	rx_drq_cnt	r/w	2'b0	RX_DRQ_CLR_CNT. When RX FIFO available data less than or equal N, DRQ Request will be de-asserted. N is defined here: 00: IRQ/DRQ de-asserted when WLEVEL <= RXTL[3:0] 01: IRQ/DRQ de-asserted when WLEVEL < 1 10: IRQ/DRQ de-asserted when WLEVEL < 2 11: IRQ/DRQ de-asserted when WLEVEL < 4 WLEVEL represents the number of valid samples in the RX FIFO
13:9	RSVD			
8	rx_ch_en	r/w	1'b0	RX_FIFO_DATIN_SRC. RX FIFO Data Input Source Select. 0: Disable 1: Enable Bit8: ADC1 data
7	RSVD			
6:5	rx_data_res	r/w	2'd0	RX_SAMPLE_BITS. Receiving Audio Sample Resolution 0: 16 bits 1: 20 bits 2: 24 bits 3: Reserved
4	rx_drq_en	r/w	1'b0	ADC_DRQ_EN. ADC FIFO Data Available DRQ Enable. 0: Disable 1: Enable
3	rx_a_int_en	r/w	1'b0	ADC_IRQ_EN. ADC FIFO Data Available IRQ Enable. 0: Disable 1: Enable
2	rx_u_int_en	r/w	1'b0	ADC_UNDERRUN_IRQ_EN. ADC FIFO Under Run IRQ Enable 0: Disable 1: Enable
1	rx_o_int_en	r/w	1'b0	ADC_OVERRUN_IRQ_EN. ADC FIFO Over Run IRQ Enable 0: Disable 1: Enable

Bits	Name	Type	Reset	Description
0	rx_fifo_flush	w1c	1'b0	ADC_FIFO_FLUSH. ADC FIFO Flush. Write ‘1’ to flush TX FIFO, self clear to ‘0’ .

16.6.16 audadc_rx_fifo_status

Address: 0x2000ac84



Bits	Name	Type	Reset	Description
31:25	RSVD			
24	rx_a	r	1'b0	RXA. RX FIFO Available 0: No available data in RX FIFO 1: More than one sample in RX FIFO (>= 1 word)
23:20	RSVD			
19:16	rx_a_cnt	r	4'h0	RXA_CNT. RX FIFO Available Sample Word Counter
15:5	RSVD			
4	rx_a_int	r	1'b0	RXA_INT. RX FIFO Data Available Pending Interrupt 0: No Pending IRQ 1: Data Available Pending IRQ Automatic clear if interrupt condition fails.
3	RSVD			
2	rxu_int	r	1'b0	RXU_INT. RX FIFO Underrun Pending Interrupt 0: No Pending IRQ 1: FIFO Underrun Pending IRQ Write ‘1’ to clear this interrupt

Bits	Name	Type	Reset	Description
1	rxo_int	r	1'b0	RXO_INT. RX FIFO Overrun Pending Interrupt 0: No Pending IRQ 1: FIFO Overrun Pending IRQ Write ‘1’ to clear this interrupt
0	RSVD			

16.6.17 audadc_rx_fifo_data

Address: 0x2000ac88

rx_data

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

rx_data

Bits	Name	Type	Reset	Description
31:0	rx_data	r	32'h0	RX_DATA. RX Sample Host can get one sample by reading this register. The left channel sample data is first and then the right channel sample.

17.1 Overview

The EMAC module is a 10/100Mbps Ethernet Media Access Controller (Ethernet MAC) compatible with IEEE 802.3. It consists of status and control register set, RX/TX module, RX/TX buffer descriptor (BD) set, master interface, MDIO interface, and physical layer chip (PHY) interface.

The status and control register set contains the status and control bits of EMAC. As the interface with the user program, it controls data sending and receiving and reports the status.

The RX/TX module obtains data frames from the specified memory according to the control words in the RX/TX descriptor, adds preamble and CRC, and expands short frames to send them through PHY. Or, it receives data from PHY, and puts them into the specified memory according to the RX/TX BD. Relevant event flags are set after sending and receiving are completed. If the event interrupt is enabled, an interrupt request will be sent to the master for processing.

MDIO and MII/RMII interfaces communicate with PHY, including reading and writing the registers of PHY and sending and receiving data packets.

17.2 Features

- Compatible with the MAC layer defined by IEEE 802.3
- PHY supporting MII/RMII interface defined by IEEE 802.3
- Interacts with PHY through MDIO interface
- Supports 10 Mbps and 100 Mbps Ethernet
- Supports half-duplex and full-duplex
- Supports automatic flow control and control frame generation in the full-duplex mode
- Supports collision detection and retransmission in the half-duplex mode

- Supports the generation and verification of CRC
- Generates and removes data frame preamble
- Supports automatic extension of short data frames when sending
- Detects too long/short data frames (length limit)
- Transmits long data frames (> standard Ethernet frame length)
- Automatically discards data packets with over-limit retransmission times or too small frame gap
- Broadcast packet filtering
- Internal RAM for storing up to 128 BDs
- Splits and configures a data packet to multiple consecutive BDs when sending
- Various event flags sent or received
- Generates a corresponding interrupt when an event occurs

17.3 Functional Description

The composition of EMAC module is as follows.

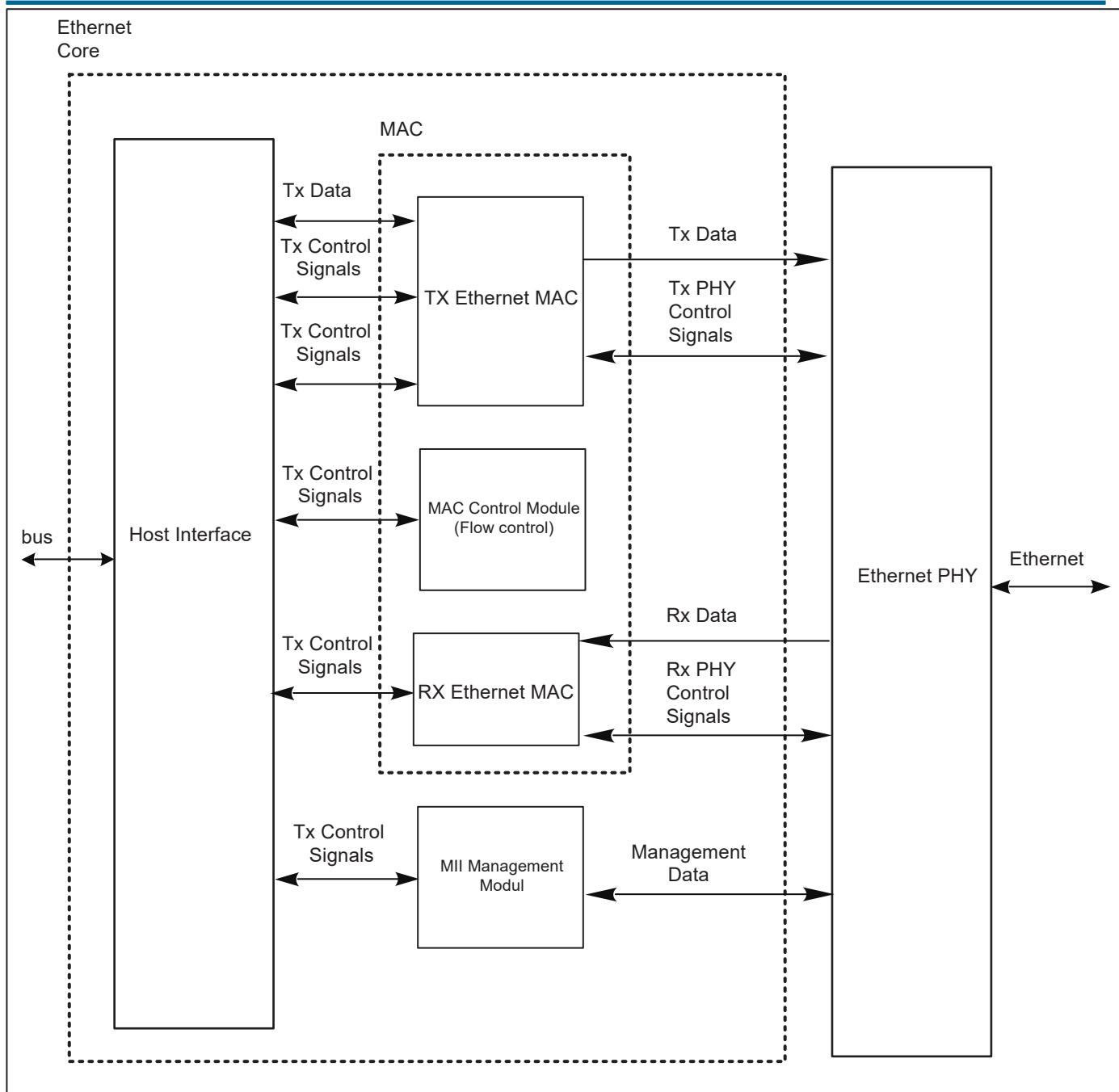


Fig. 17.1: Block diagram of EMAC

Through MDIO interface, the control register can read and write PHY's registers, to perform configuration, mode selection (half/full duplex), negotiation, and other operations. The RX module filters and checks the received data frames for a valid preamble, FCS, and length, and stores the data in the specified memory address according to the BD. The TX module gets data from the memory according to the data BD, adds preamble, FCS, and pad, and then sends them out using the CSMA/CD protocol. If CRS is detected, retry will be delayed. The RX/TX BD set is connected to the system RAM, which is used to store the sent and received Ethernet data frames. Each descriptor contains the corresponding control status word and buffer memory address. There are 128 descriptors for RX/TX, and can be allocated flexibly.

17.4 Clock

EMAC needs a clock for synchronous transmission and reception (25 MHz (MII) or 50 MHz (RMII) at 100 Mbps, and 2.5 MHz at 10 Mbps). The clock must be synchronized between EMAC and PHY.

17.5 RX/TX BD

The RX/TX BD provides the association between EAMC and data frame cache address information, controls RX/TX data frames, and gives RX/TX status prompt. Each descriptor consists of two consecutive words (1 word = 32 bits). The word0 with low address provides the length, control bits, and status bits of the data frames contained in this buffer. The word1 with high address is the memory pointer.

Specific description of word0 in TX BD:

[31:16]: TX packet length (LEN).

[15]: TX BD Ready (RD) flag. The software writes “1” to inform EMAC that this BD contains data to be sent, and the hardware writes “0” to indicate that the BD data has been sent or an error has occurred.

[14]: Interrupt Request (IRQ) flag bit. When set, this BD can request TXE or TXB interrupt.

[13]: Wraparound (WR) flag. When set, it indicates that this BD is the last TX BD, and the hardware sends it again from the starting BD.

[12]: Padding (PAD). When it is set and a padding permission is set in EMAC, TX BD automatically fills the too short packet.

[11]: Cyclic Redundancy Check (CRC). When set, EMAC automatically calculates the CRC of the sent packet and attach it to the packet.

[10]: End of Frame (EoF) flag. If a frame of data occupies multiple BDs, this bit marks the end of this frame of data.

[8]: Underrun (UR) flag. When set, it indicates that the FIFO underrun error occurred during BD transmission.

[7:4]: Retry (RTRY) times counter. It counts the retry times.

[3]: Retry Limit (RL) flag. When set, it indicates that the retry times exceed the maximum retry times (MAXRET) configured in COLLCNF.

[2]: Late Collision (LC) flag. When set, it indicates that late collision occurred when this BD is sent.

[1]: Defer Indication (DF) flag. When set, it indicates that this packet is delayed.

[0]: Carrier Sense (CS) failure. If no carrier is detected during sending, it is set.

Specific description of word0 in RX BD:

[31:16]: TX packet length (LEN).

[15]: RX BD (RD) empty flag. When set, it indicates that this BD is empty (no received data is saved). “Clearing”

indicates that this BD has received data or an error occurred during receiving.

[14]: Interrupt Request (IRQ) flag bit. When set, this BD can request RXE or RXB interrupt.

[13]: Wraparound (WR) flag. When set, it indicates that the BD is the last RX BD, and the hardware sends it again from the starting BD.

[8]: Control Frame (CF) flag. When set, it indicates that this BD has received one Control Frame.

[7]: Miss (M) flag. If a packet is received in promiscuous mode but it is marked as Miss by the internal address logic, EMAC sets this flag bit.

[6]: Overrun (OR) flag. When set, it indicates that the FIFO overrun error occurred during receiving.

[5]: Receive Error (RE) flag. When set, it indicates that the RX ERR signal sent by PHY is received during receiving.

[4]: Dribble Nibble (DN) flag. When set, it indicates that an odd number of nibbles have been received.

[3]: Too Long (TL) packet flag. When set, it indicates that the received packet is too long, exceeding the setting value.

[2]: Too Short (SF) packet flag. When set, it indicates that the received packet is shorter than the minimum allowable length.

[1]: CRC (CRC) error flag. When set, it indicates that the CRC of the received packet fails.

[0]: Late Collision (LC) flag. When set, it indicates that Late Collision occurred when data is received to this BD.

It should be noted that BD must be written by word. EMAC supports 128 BDs, which are shared by the RX/TX logic and can be freely combined. But the TX BD always occupies the preceding contiguous area. The number of BD is specified by the TXBDNUM field in the register MAC_TX_BD_NUM. EMAC circularly processes the RX/TX BDs according to their order, until it finds the BDs marked WR, and goes back to the first RX/TX BD respectively.

17.6 PHY Interaction

The interactive register set of PHY provides a way to communicate commands and data needed for interaction with PHY. EMAC controls the working mode of PHY through MDIO interface, and ensures the matching of both working modes (such as rate, full/half-duplex). Data packets interact between EMAC and PHY through MII/RMII interface, which is selected by the RMII_EN bit in the EMAC's mode register (EMAC_MODE): When this bit is 1, RMII mode is selected, and otherwise the MII mode is selected. Both MII and RMII modes support the transmission rates of 10 Mbps and 100 Mbps specified in IEEE 802.3u. The transmission signals of MII and RMII are described as follows.

Table 17.1: Transmission signal

Name	MII	RMII
EXTCK_EREFCK	ETXCK: Send clock signal	EREFCK:reference clock
ECRS	ECRS: carrier detection	-
ECOL	ECOL: collision detection	-
ERXDV	ERXDV: valid data	ECRSDV: carrier detection/valid data

Table 17.1: Transmission signal(continued)

Name	MII	RMII
ERX0-ERX3	ERX0ERX3: 4-bit received data	ERX0ERX1: 2-bit received data
ERXER	ERXER: Receive error indication	ERXER: Receive error indication
ERXCK	ERXCK: Receive clock signal	-
ETXEN	ETXEN: TX enable	ETXEN: TX enable
ETX0-ETX3	ETX0ETX3: 4-bit sent data	ETX0ETX1: 2-bit sent data
ETXER	ETXER: Send error indication	-
EMDC	MDIO Clock	MDIO Clock
EMDIO	MDIO Data Input Output	MDIO Data Input Output

The RMII interface has fewer pins, and 2-bit data lines are used for transmission and reception. At 100 Mbps, a reference clock of 50 MHz is required.

17.7 Programming Flow

17.7.1 PHY initialization

- Select a proper connection mode by setting the RMII_EN bit in the register EMAC_MODE according to the PHY type.
- Set the MAC address of EMAC to EMAC_MAC_ADDR0 and EMAC_MAC_ADDR1
- Set an appropriate clock for the MDIO part by programming the CLKDIV field in the register EMAC_MIIMODE
- Set the address of the corresponding PHY to the FIAD field of the register EMAC_MIIADDRESS
- According to PHY's manual, send commands through registers EMAC_MIICOMMAND and EMAC_MIITX_DATA
- Store the data read from PHY in the register EMAC_MIIRX_DATA
- Query the status of interaction with PHY commands through the register EMAC_MIISTATUS

After basic interaction is completed, PHY shall be switched to the auto-negotiation state. Upon negotiation completed, the mode must be programmed to the FULLD bit in the EMAC_MODE register based on the negotiation result.

17.7.2 Send Data Frame

- Configure data frame format and interval bit fields in the register EMAC_MODE
- Specify the number of TX BDs by setting the TXBDNUM field in the register EMAC_TX_BD_NUM, so that the rest is the number of RX BDs
- Prepare the data frames to be sent in the memory
- Fill in the address of data frames in the data pointer field (word 1) corresponding to the TX BDs
- Clear the status flag in the control and status fields (word 0) corresponding to the TX BDs, and set the control field (CRC enable, PAD enable, and interrupt enable)
- Write the data frame length, and set the RD field to inform EMAC that this BD data needs to be sent. If necessary, set the upper IRQ bit to enable interrupt
- Especially, if it is the last TX BD, the upper WR bit must be set. EMAC will "go back" to the first TX BD after this BD is processed
- If there are multiple BDs to be sent, repeat the steps of setting BD to pad all the TX BDs
- If one packet is contained in only one BD, set its EOF bit to 1
- If one packet is sent in multiple BDs, just mark the last BD it occupies as the end of the packet by setting the EOF bit
- To enable the TX interrupt, configure the TX-related bits in the register EMAC_INT_MASK
- Configure the TXEN bit in the register EMAC_MODE to enable TX
- If an interrupt is enabled, in the TX interrupt, obtain the current BD through the TXBDNUM field in the register EMAC_TX_BD_NUM
- Complete processing based on the status word of the current BD
- For BDs whose data has been sent, the RD bit in its control field will be cleared by hardware and BDs will not be used for TX again. Only after new data is padded and RD is set, can this BD be used for TX again

17.7.3 Receive Data Frame

- Configure data frame format and interval bit fields in the register EMAC_MODE
- Specify the number of TX BDs by setting the TXBDNUM field in the register EMAC_TX_BD_NUM, so that the rest is the number of RX BDs
- Prepare an area in memory for receiving data
- Fill in the address of data frames in the data pointer field (word 1) corresponding to the RX BDs
- Clear the status flag in the control and status fields (word 0) corresponding to the RX BDs, and set the control field

(interrupt enable)

- Write the receivable data frame length, and set the E-bit field to inform EMAC that this BD is free and can receive data. If necessary, set the upper IRQ bit to enable the interrupt
- Especially, if it is the last valid RX BD, the upper WR bit must be set. EMAC will "go back" to the first RX BD after this BD is processed
- If there are multiple BDs for receiving data, repeat the steps of setting BD to pad all the BDs
- To enable the RX interrupt, configure the RX-related bits in the register EMAC_INT_MASK
- Configure the RXEN bit in the register EMAC_MODE to enable RX
- If an interrupt is enabled, in the RX interrupt, obtain the current BD through the RXBDNUM field in the register EMAC_TX_BD_NUM
- Complete processing based on the status word of the current BD
- For BDs whose data has been received, the E bit in its control field will be cleared by hardware and BDs will not be used for RX again. Only after you take out the data and set the E bit, can this BD be used for RX again

18

USB

18.1 Overview

19.1 Overview

ISO11898 was developed by German BOSCH Company, which is famous for R&D and production of automotive electronic products. It is one of the most widely used fieldbuses in the world.

19.2 Features

- Supports 1Mbps and custom bit rate
- ISO11898 2.0A and 2.0B
- Self-test mode (self-sending and self-receiving)
- Frame filtering
- Silent mode (no reply, no valid error flag)
- Single transmission without retransmission
- Query of bits that lose the arbitration
- Any bus error can trigger an interrupt

19.3 Functional Description

19.3.1 TX Buffer (TXB)

TXB, an interface between CPU and BSP, can store complete messages for transmission over the ISO11898 network. The buffer length is 13 bytes, written by CPU and read by the BSP.

19.3.2 RX Buffer (RXB, RXFIFO)

RXB, an interface between access control filter and CPU, is used to store the filtered messages received from the ISO11898 bus. RXB represents a 13-byte window in the RX FIFO that can be accessed by CPU, with a total length of 64 bytes. RX FIFO allows CPU to receive other messages while processing one frame of message.

19.3.3 Access Control Filter (ACF)

ACF compares the received identifier with the contents of the ACF register and decides whether the message should be accepted. If accepted, the complete message will be stored in RX FIFO.

19.3.4 Bit Stream Processor (BSP)

BSP, a sequence generator, is used to process the data between TXB, RXB, and ISO11898 bus. It also performs error detection, arbitration, bit padding, and error handling on the ISO11898 bus.

19.3.5 Bit Timing Logic (BTL)

BTL monitors the serial ISO11898 bus and processes the bus related bit timing. It performs hard synchronization on the recessive-to-dominant transition of the bus at the beginning of the message, and soft synchronization again during subsequent message reception. BTL also provides a programmable time period to compensate propagation delay and phase shift (for example, due to oscillator drift), and can define sampling points and sampling times in a bit time.

19.3.6 Error Management Logic (EML)

EML determines the errors of the transport layer module. It receives the error statement from BSP and then informs BSP and interrupt management logic (IML) of error statistics.

19.4 Functional Description

19.4.1 Mode

19.4.1.1 Self-test Mode

You can select the self-test mode by setting the STM bit in the MOD register to ‘1’ . In this mode, the RX request command can be used to conduct a full-node test without other active nodes on the bus, and even if no response is received, the ISO11898 controller will perform successful transmission.

19.4.1.2 Silent Mode

You can select the silent mode by setting the LOM bit in the MOD register to “1”. In this mode, the ISO11898 controller will not respond to the ISO11898 bus even if it successfully receives the message, and the error counter will stay at the current value. This mode will force the ISO11898 controller to become a passive error, and no message can be transmitted at this time. This mode can be used in software-driven bit rate detection and hot swap scenarios, and all other functions work normally as the normal mode.

19.4.1.3 Reset Mode

Once the RM bit in the MOD register changes from ‘0’ to ‘1’, it will cause the current TX and RX messages to be terminated and enter the reset mode. When the RM bit changes from ‘1’ to ‘0’, the ISO11898 controller will return to the operating mode.

The meanings of different operations in different modes are as follows.

Table 19.1: The meaning of each register in different modes

ADDRESS OFFSET	OPERATING MODE		RESET MODE	
	READ	WRITE	READ	WRITE
0x00	mode	mode	mode	mode
0x04	(00H)	command	(00H)	command
0x08	status	reserved	status	reserved
0x0C	interrupt	reserved	interrupt	reserved
0x10	interrupt enable	interrupt enable	interrupt enable	interrupt enable
0x14	reserved	reserved	reserved	reserved
0x18	bus timing 0	reserved	bus timing 0	bus timing 0
0x1C	bus timing 1	reserved	bus timing 1	bus timing 1
0x20	reserved	reserved	reserved	reserved
0x24	reserved	reserved	reserved	reserved
0x28	reserved	reserved	reserved	reserved
0x2C	arbitration lost capture	reserved	arbitration lost capture	reserved
0x30	error code capture	reserved	error code capture	reserved
0x34	error warning limit	reserved	error warning limit	error warning limit
0x38	RX error counter	reserved	RX error counter	RX error counter
0x3C	TX error counter	reserved	TX error counter	TX error counter
0x40	SFF RX frame information	EFF RX frame information	SFF TX frame information	EFF TX frame information
			acceptance code 0	acceptance code 0

Table 19.1: The meaning of each register in different modes(continued)

ADDRESS OFFSET	OPERATING MODE				RESET MODE	
	READ		WRITE		READ	WRITE
0x44	RX identifier 1	RX identifier 1	TX identifier 1	TX identifier 1	acceptance code 1	acceptance code 1
0x48	RX identifier 2	RX identifier 2	TX identifier 2	TX identifier 2	acceptance code 2	acceptance code 2
0x4C	RX data 1	RX identifier 3	TX data 1	TX identifier 3	acceptance code 3	acceptance code 3
0x50	RX data 2	RX identifier 4	TX data 2	TX identifier 4	acceptance mask 0	acceptance mask 0
0x54	RX data 3	RX data 1	TX data 3	TX data 1	acceptance mask 1	acceptance mask 1
0x58	RX data 4	RX data 2	TX data 4	TX data 2	acceptance mask 2	acceptance mask 2
0x5C	RX data 5	RX data 3	TX data 5	TX data 3	acceptance mask 3	acceptance mask 3
0x60	RX data 6	RX data 4	TX data 6	TX data 4	reserved	reserved
0x64	RX data 7	RX data 5	TX data 7	TX data 5	reserved	reserved
0x68	RX data 8	RX data 6	TX data 8	TX data 6	reserved	reserved
0x6C	(FIFO RAM)	RX data 7	reserved	TX data 7	reserved	reserved
0x70	(FIFO RAM)	RX data 8	reserved	TX data 8	reserved	reserved
0x74	RX message counter		reserved		RX message counter	reserved
0x78	RX buffer start address		reserved		RX buffer start address	RX buffer start address
0x7C	clock divider		clock divider		clock divider	clock divider

19.4.2 Sending Process

19.4.2.1 Process

1. Check the TBS bit in the SR register to ensure that TXB is empty.
2. Configure frame information, ID number, and data.
3. Request transmission by setting the TR bit in the CMR register.

19.4.2.2 Termination of Sending

When CPU requests to suspend the previous transmission, you can use this function. For example, you need to send a more urgent message first. Messages that are sending now are not affected by this function and sending will not stop. To check whether the previous message was successfully sent, you should check the TCS bit in the SR register. The application software can use this function by setting the AT bit in the CMR register to ‘1’ , which should be executed after the TBS bit in the SR register is set to ‘1’ or the TX interrupt is generated.

It should be noted that even if the message is terminated, a TX interrupt will occur, because the status bit of the TXB has indicated the “released” status.

19.4.2.3 Self-sending and Self-receiving

The application software can realize self-sending and self-receiving by setting the SRR bit in the CMR register. At that time, sending and receiving are synchronized. Other operations are the same as the normal sending process.

19.4.2.4 Precautions

1. If the TR and AT bits of the CMA register are set simultaneously, the message will be sent only once. Even if there is an error event or arbitration lost, it will not be sent again.
2. If the SRR and AT bits of the CMA register are set simultaneously, the message will be sent only once by self-sending and self-receiving. Even if there is an error event or arbitration lost, it will not be sent again.
3. If SRR, TR, and AT bits of the CMA register are set simultaneously, the message will be sent by setting TR and AT bits simultaneously.
4. Once the TX status bit in the status register is set, the internal TX request bit will be cleared automatically.
5. If the TR and SRR bits of the CMA register are set simultaneously, the SRR bit will be ignored.

19.4.3 Receiving Process

19.4.3.1 Process

The received messages are stored in an internal FIFO with a depth of 64 bytes. The FIFO is completely managed by hardware, which saves CPU’s processing load, simplifies software, and ensures data consistency. The application can read the received messages through the FIFO’s output interface. When the RBS bit in the SR register is set, one or more frames of messages can be read in RX FIFO. After the software gets the message, setting the RRB bit in the CMR register can release the RX FIFO occupied by the current message.

19.4.3.2 Number of Messages

The RMC register indicates the number of readable messages in RX FIFO, which increases with each RX event and decreases with each buffer release. The value is 0 after reset.

19.4.3.3 RXB

The RBSA register indicates the address of the first byte of the received message stored in the current internal RAM, which is mapped to the RXB window. The contents of the internal RAM can be interpreted on this basis. This part of the internal RAM can be read and written by CPU (written only in the reset mode).

Example: If the value of RBSA is 18H, the current readable message of the RXB window (offset address: 10H to 12H) is also stored in the RAM address starting from 18H. As the RAM address is directly mapped to the starting position of ISO11898 offset address 20H (corresponding to RAM address 0H), the message can also be read from ISO11898 offset address 38H and the following bytes (ISO11898 address = RBSA + 20H = 18H + 20H = 38H). If the message address exceeds the RAM address 3FH, it will continue from the RAM address 0.

When there is at least one message in FIFO, the command to release the RXB should be issued, and then RBSA will be updated to the starting position of the next message.

When the hardware is reset, the value of RBSA register is initialized to ‘00H’ . When the software is reset in the reset mode, the value of this register will not change, but the FIFO will be cleared. This means that the contents of RAM will not change, but the next received (or sent) message will overwrite the visible message in the RXB window.

19.4.4 Identifier Filtering

With the help of ACF, the ISO11898 controller will allow the received message to be delivered to RX FIFO only when the identifier bit of the received message is the same as the predefined bit in the ACF register. ACF consists of the acceptance code registers (ACRn) and acceptance mask registers (AMRn). The values of matching bits in the receivable message are set by the ACRn, and which bits can be masked is set by the AMRn.

There are two different filtering modes (set by the AFM bit in the MOD register):

- Single filter mode (AFM = 1).
- Double filter mode (AFM = 0).

19.4.4.1 Single Filter Configuration

In this configuration, a 4-byte long filter can be defined. The bit correspondence between filter bytes and message bytes depends on the currently received frame format.

Standard frame: If a message in a standard frame format is received, the complete identifier including the RTR bit and the first two data bytes is used to accept filtering. If there is no data byte because RTR bit is set, or there is no data byte or only one data byte because a data length is set, the message can also be received.

As all the filter bits are logically AND, only when all the bits pass through the filter, can a message be received. It should be noted that the low 4 bits of AMR1 and ACR1 are unused, and these bits should be set as mask bits for compatibility with future products. That is, all 3-0 bits of AMR1 are ‘1’ .

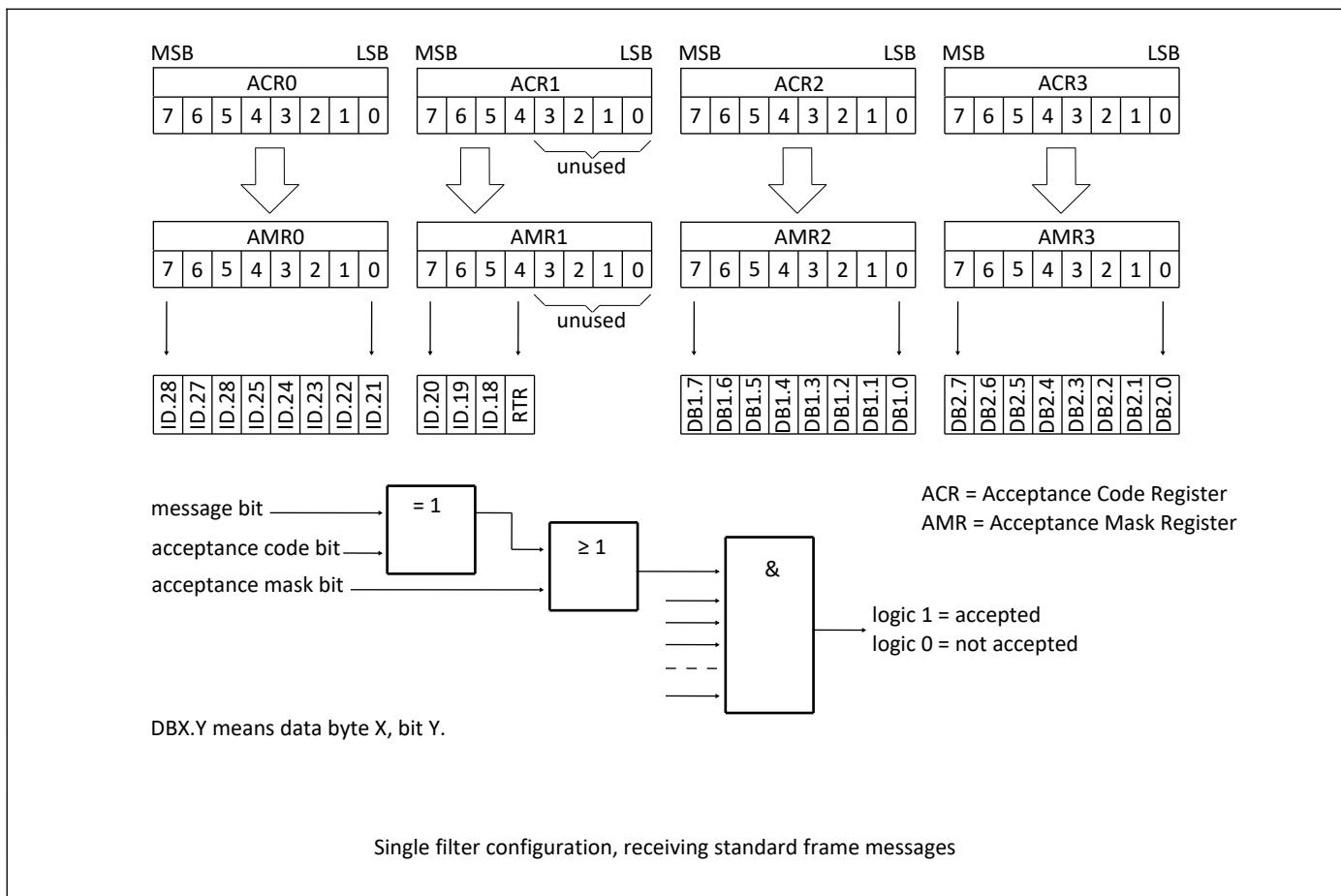


Fig. 19.1: Single filter configuration, receiving standard frame messages

Extended frame: If a message in an extended frame format is received, the complete identifier including the RTR bit is used to accept filtering.

As all the filter bits are logically AND, only when all the bits pass through the filter, can a message be received. It should be noted that the low 2 bits of AMR3 and ACR3 are unused, and these bits should be set as mask bits for compatibility with future products. That is, all 1-0 bits of AMR3 are ‘1’ .

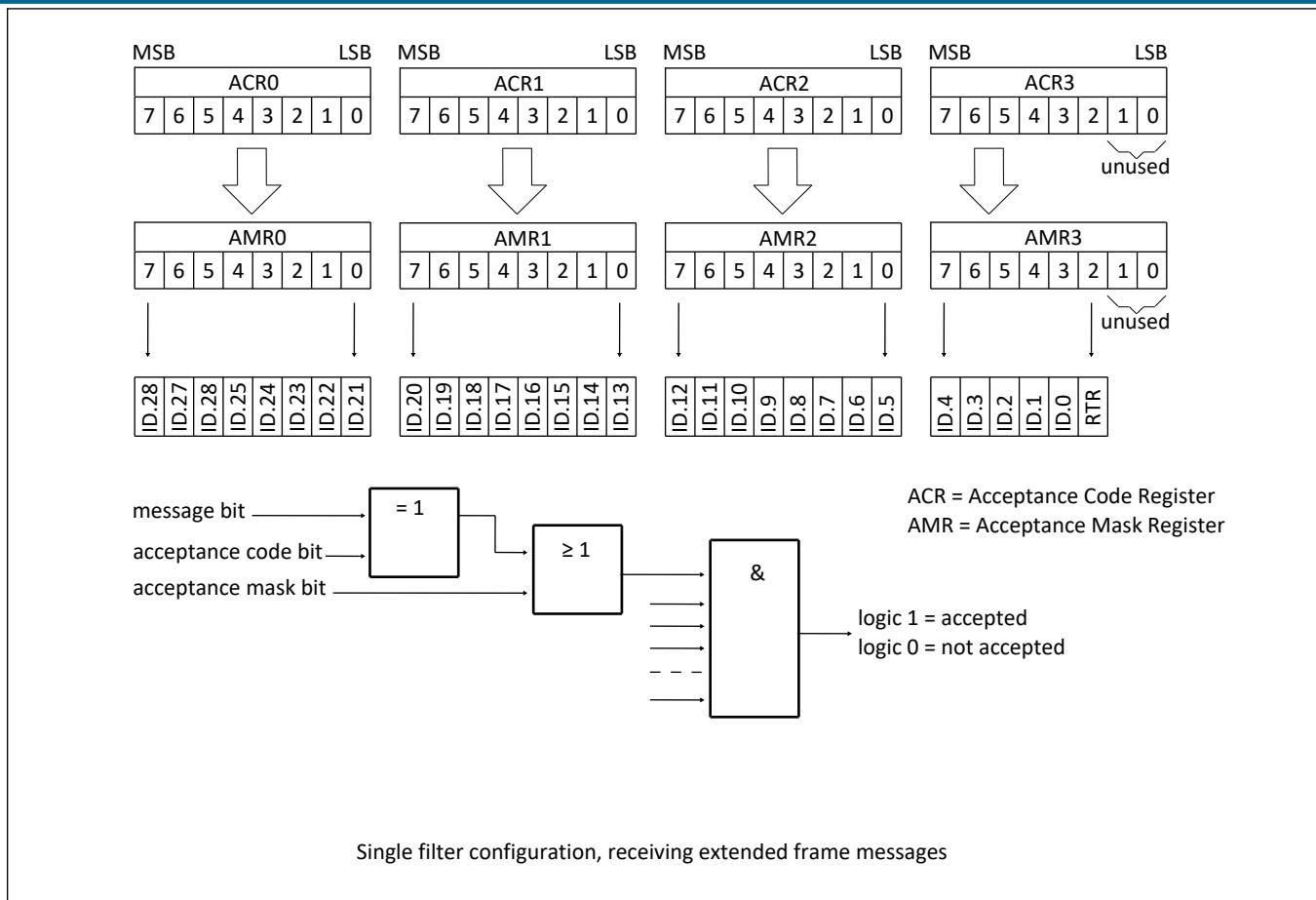


Fig. 19.2: Single filter configuration, receiving extended frame messages

19.4.4.2 Double Filter Configuration

Two short filters can be defined in this configuration, and the received message will be compared with both filters to decide whether to copy the message to the RXB. As long as a filter receives the message, the received message is valid. The bit correspondence between filter bytes and message bytes depends on the currently received frame format.

Standard frame: If a message in a standard frame format is received, the two filters defined look a little different. The first filter compares the complete identifier including RTR and the first data byte, while the second one only compares the standard identifier including RTR.

To successfully receive the message, the comparison result of all single bits in at least one complete filter indicates “accept”. There is no data when RTR is set or the data length is 0. However, if the first part up to the RTR bit indicates “accept”, the message can also pass through the filter 1.

If the first filter does not need to filter data bytes, the low 4 bits of AMR1 and AMR3 must be set to logical ‘1’ (insignificant), and the two filters run identically using standard identifiers including RTR.

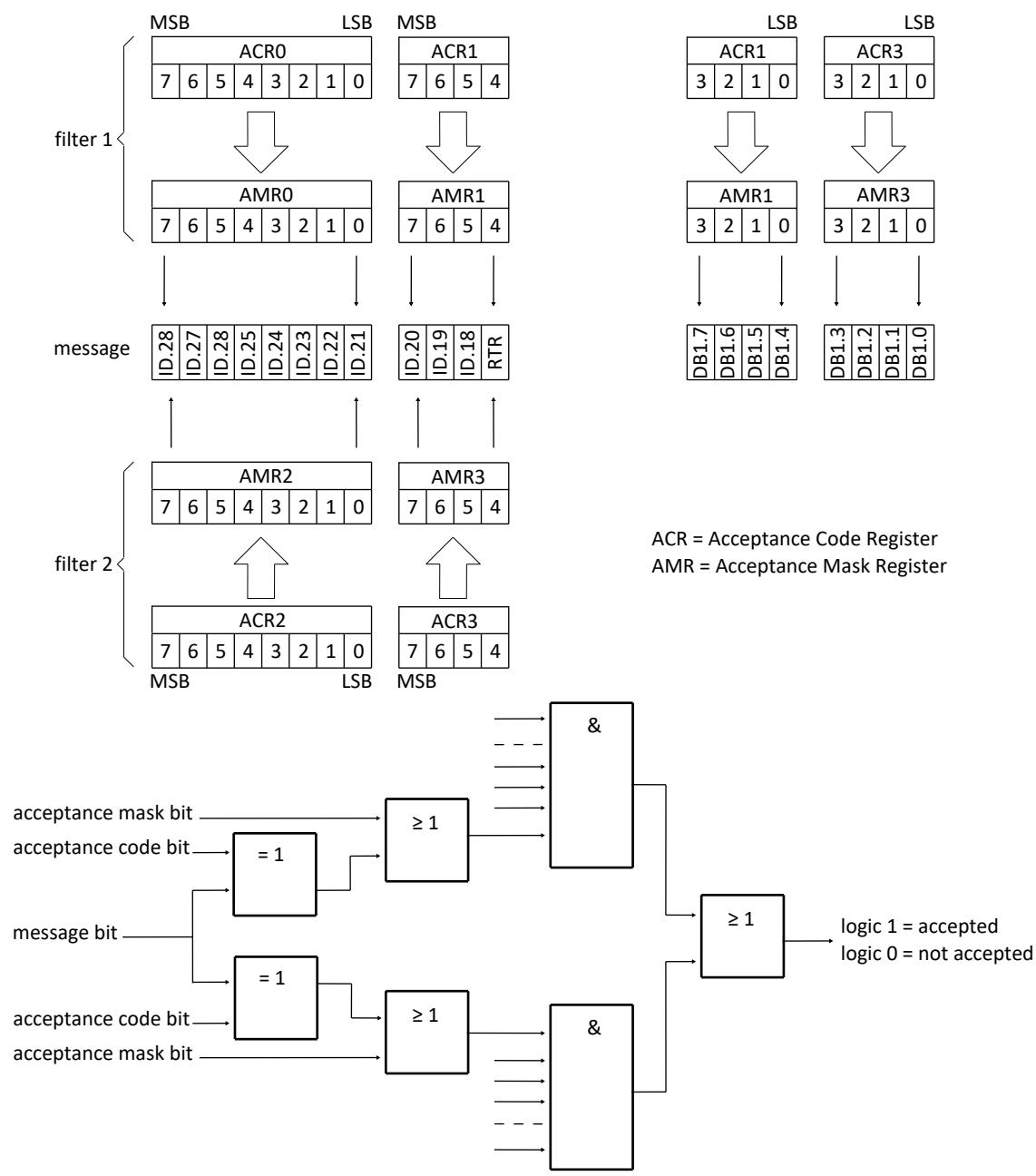


Fig. 19.3: Dual filter configuration, receiving standard frame messages

Extended frame: If a message in an extended frame format is received, the two filters defined look the same. Both filters only compare the first two bytes of the extended identifier.

Only when all single bit comparisons of at least one complete filter indicate acceptance, can the message be suc-

cessfully received.

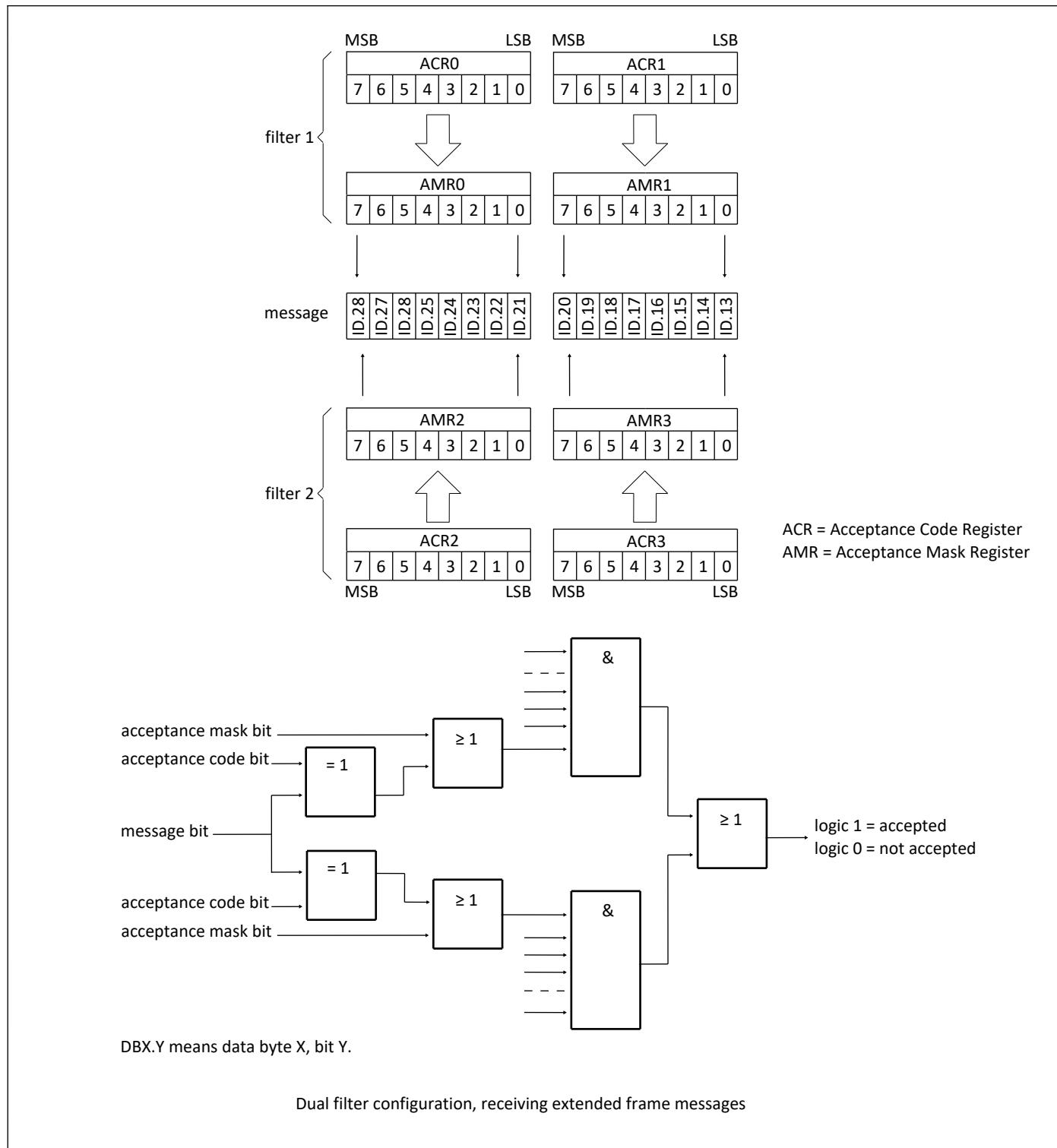


Fig. 19.4: Dual filter configuration, receiving extended frame messages

19.4.5 Error Management

19.4.5.1 Arbitration Lost

The arbitration lost capture (ALC) register contains the position that encounters arbitration lost and can only be read by CPU but not written by it. If the arbitration lost interrupt is enabled, an interrupt will be generated once arbitration loses. The position of the current bit in BSP is captured into the ALC. This register's value is fixed until the user software reads the contents of the ALC. After this value is read, the capture mechanism is activated again. When the interrupt register is read, the corresponding interrupt flag will also be cleared. No arbitration lost interrupt will be generated again before the ALC register is read.

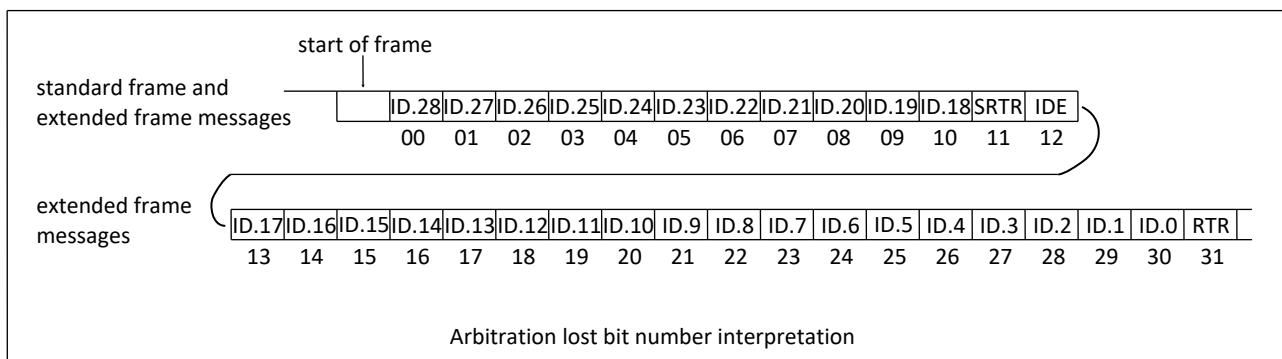


Fig. 19.5: Arbitration lost bit number interpretation

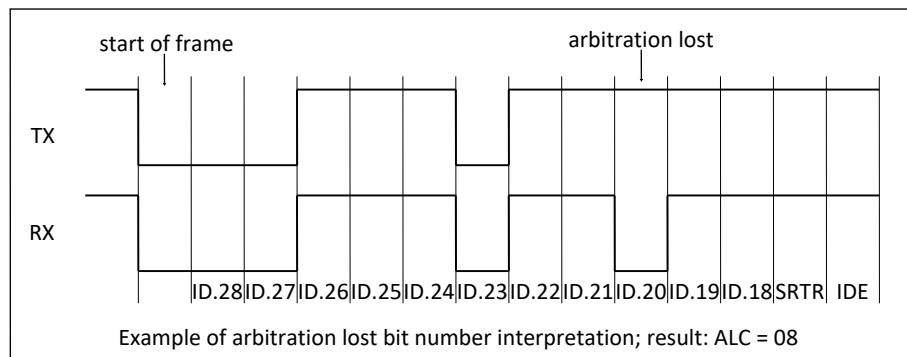


Fig. 19.6: Example of arbitration lost bit number interpretation; result: ALC = 08

Table 19.2: Arbitration loss capture location

BITS					DECIMAL VALUE	FUNCTION
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	0	0	0	00	arbitration lost in bit 1 of identifier
0	0	0	0	1	01	arbitration lost in bit 2 of identifier
0	0	0	1	0	02	arbitration lost in bit 3 of identifier
0	0	0	1	1	03	arbitration lost in bit 4 of identifier

Table 19.2: Arbitration loss capture location(continued)

BITS					DECIMAL VALUE	FUNCTION
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	1	0	0	04	arbitration lost in bit 5 of identifier
0	0	1	0	1	05	arbitration lost in bit 6 of identifier
0	0	1	1	0	06	arbitration lost in bit 7 of identifier
0	0	1	1	1	07	arbitration lost in bit 8 of identifier
0	1	0	0	0	08	arbitration lost in bit 9 of identifier
0	1	0	0	1	09	arbitration lost in bit 10 of identifier
0	1	0	1	0	10	arbitration lost in bit 11 of identifier
0	1	0	1	1	11	arbitration lost in bit SRTTR
0	1	1	0	0	12	arbitration lost in bit IDE
0	1	1	0	1	13	arbitration lost in bit 12 of identifier
0	1	1	1	0	14	arbitration lost in bit 13 of identifier
0	1	1	1	1	15	arbitration lost in bit 14 of identifier
1	0	0	0	0	16	arbitration lost in bit 15 of identifier
1	0	0	0	1	17	arbitration lost in bit 16 of identifier
1	0	0	1	0	18	arbitration lost in bit 17 of identifier
1	0	0	1	1	19	arbitration lost in bit 18 of identifier
1	0	1	0	0	20	arbitration lost in bit 19 of identifier
1	0	1	0	1	21	arbitration lost in bit 20 of identifier
1	0	1	1	0	22	arbitration lost in bit 21 of identifier
1	0	1	1	1	23	arbitration lost in bit 22 of identifier
1	1	0	0	0	24	arbitration lost in bit 23 of identifier
1	1	0	0	1	25	arbitration lost in bit 24 of identifier
1	1	0	1	0	26	arbitration lost in bit 25 of identifier
1	1	0	1	1	27	arbitration lost in bit 26 of identifier
0	1	1	0	0	28	arbitration lost in bit 27 of identifier
1	1	1	0	1	29	arbitration lost in bit 28 of identifier
1	1	1	1	0	30	arbitration lost in bit 29 of identifier
1	1	1	1	1	31	arbitration lost in bit RTR

19.4.5.2 Error Capture

The error code capture (ECC) register contains the type and location of bus errors and can only be read by CPU but not written by it. If the bus error interrupt is enabled, a bus error interrupt will be generated once a bus error occurs. The position of the current bit in BSP is captured into the ECC. This register's value is fixed until the user software reads the contents of the ECC. After this value is read, the capture mechanism is activated again. Reading the corresponding bit in the interrupt register will clear this bit, and no bus error interrupt will be generated before the ECC register is read.

The error types represented by the values in the ECC register are shown as follows.

Table 19.3: Type of error catch

BIT ECC.7	BIT ECC.6	FUNCTION
0	0	bit error
0	1	form error
1	0	stuff error
1	1	other type of error

Table 19.4: Error catch location

BIT ECC.4	BIT ECC.3	BIT ECC.2	BIT ECC.1	BIT ECC.0	FUNCTION
0	0	0	1	1	start of frame
0	0	0	1	0	ID.28 to ID.21
0	0	1	1	0	ID.20 to ID.18
0	0	1	0	0	bit SRTR
0	0	1	0	1	bit IDE
0	0	1	1	1	ID.17 to ID.13
0	1	1	1	1	ID.12 to ID.5
0	1	1	1	0	ID.4 to ID.0
0	1	1	0	0	bit RTR
0	1	1	0	1	reserved bit 1
0	1	0	0	1	reserved bit 0
0	1	0	1	1	data length code
0	1	0	1	0	data field
0	1	0	0	0	CRC sequence
1	1	0	0	0	CRC delimiter
1	1	0	0	1	acknowledge slot

Table 19.4: Error catch location(continued)

BIT ECC.4	BIT ECC.3	BIT ECC.2	BIT ECC.1	BIT ECC.0	FUNCTION
1	1	0	1	1	acknowledge delimiter
1	1	0	1	0	end of frame
1	0	0	1	0	intermission
1	0	0	0	1	active error flag
1	0	1	1	0	passive error flag
1	0	0	1	1	tolerate dominant bits
1	0	1	1	1	error delimiter
1	1	1	0	0	overload flag

19.4.5.3 RX Error Counter Register (RXERR)

The RXERR's value represents the current number of received errors, and this register is initialized to logical '0' after hardware reset. In the operating mode, this register can only be read by CPU and write in the reset mode. RXERR is set to logical '0' if a bus shutdown event occurs. At this time, the bus is OFF, and the write operation to this register does not work.

It should be noted that CPU can modify the value of RXERR only in the reset mode. In this case, the error state may change, and the error warning interrupt and error passive interrupt will not occur unless the reset mode is exited.

19.4.5.4 TX Error Counter Register (TXERR)

The TXERR's value represents the current number of send errors. In the operating mode, this register can only be read by CPU and write in the reset mode. This register's value is initialized to logical '0' after hardware reset. If a bus shutdown event occurs, the TXERR's value is set to 127, so that the shortest time (128 bus idle signals) defined by the protocol can be calculated. Reading the TXERR's value during this period can obtain the status information of bus shutdown recovery. If the bus is OFF, a write to TXERR ranging from 0 to 254 will clear the Bus Off flag, and the controller will wait for 11 consecutive recessive bits (Bus Idle) to appear once after clearing the reset mode.

Writing 255 into TXERR by CPU will generate a bus shutdown event. It should be noted that CPU can only change the value of this register by force in the reset mode. In this case, the error or bus state may change, and the error warning interrupt or error passive interrupt will not be affected by the new value unless the reset mode is exited again. After the reset mode is exited, the TXERR's value still operates as if the bus was shut down due to a bus error, which means that the register will enter the reset mode again, the TXERR's value is initialized to 127, the RXRERR's value is initialized to 0, and related statuses and interrupt registers are reset. At this time, exiting the reset mode will execute the bus shutdown recovery process defined by the protocol (waiting for 128 bus idle signals). If the reset

mode is enabled again before the bus is turned off and restored ($\text{TXERR} > 0$), the bus will remain OFF and the TXERR 's value will be frozen.

19.4.5.5 Error Limit Setting

The error warning limit can be set by the EWLR register, whose default value is 96 (after hardware reset). This register can be read or written by CPU in the reset mode, while it can only be read in the operating mode. When at least one of the two error count values from RXERR and TXERR is greater than or equal to the value set in the EWLR register, the ES bit in the SR register will be set, and otherwise it will be cleared. Then, if the EIE bit in the IER register is set, an error warning interrupt will be generated. It is worth noting that this register can only be operated in the reset mode. The operation of this register may cause the error state to change, and the error warning interrupt will not be generated unless the reset mode is exited again.

19.4.6 Bit Timing

The timing diagram is as follows:

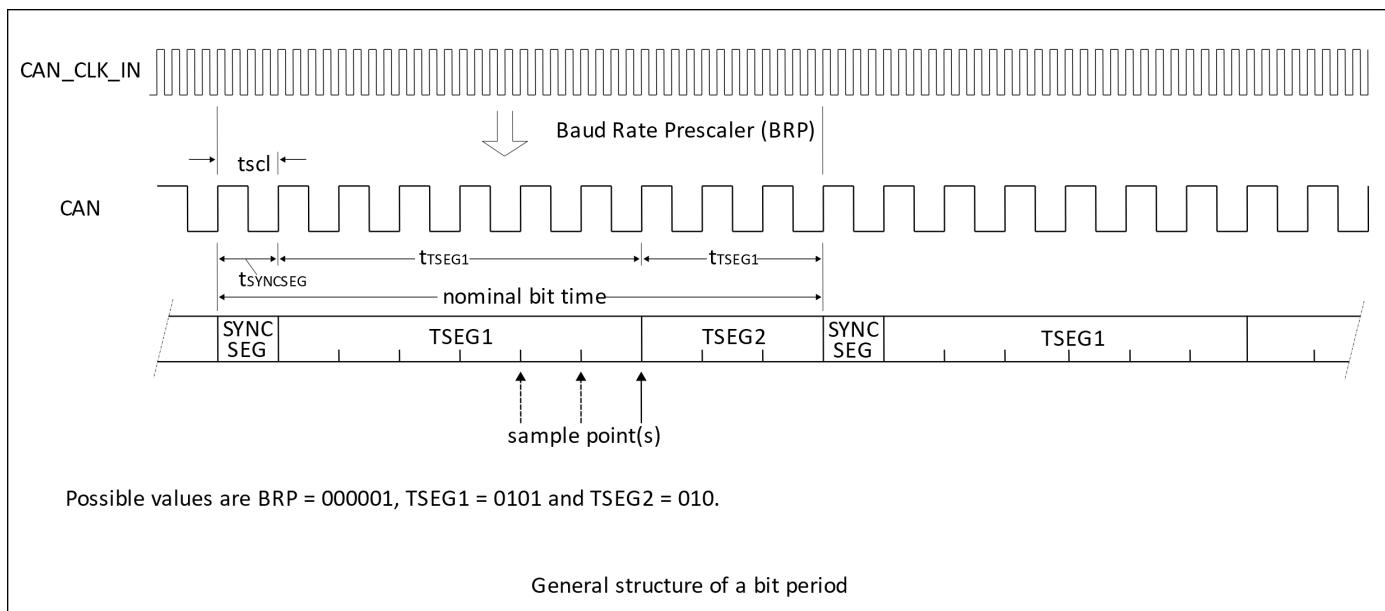


Fig. 19.7: General structure of a bit period

19.4.6.1 Baud Rate Prescaler (BRP)

The cycle of the system clock tscl of the ISO11898 controller can be set, and this determines the timing of each bit. The calculation formula of ISO11898 system clock is as follows:

$$\text{tscl} = 2 * \text{tCLK} * (32 * \text{BRP.5} + 16 * \text{BRP.4} + 8 * \text{BRP.3} + 4 * \text{BRP.2} + 2 * \text{BRP.1} + \text{BRP.0} + 1)$$

19.4.6.2 Synchronization Jump Width (SJW)

To compensate the phase shift between the clock oscillators of different bus controllers, any bus controller must resynchronize at the edge of any related signal currently being transmitted. SJW defines the maximum number of clock cycles that a bit cycle can shorten or extend through one resynchronization:

$$t_{SJW} = tscl * (2 * SJW.1 + SJW.0 + 1)$$

19.4.6.3 Sampling (SAM)

When the SAM bit in the BTR1 register is 1, the bus will be sampled three times. This mode suits medium and low speed buses, and this is beneficial to the filter in the bus. If the SAM bit is 0, the bus will only be sampled once. This mode suits the high-speed bus.

19.4.6.4 Time Segment (TSEG)

TSEG, consisting of TSEG1 and TSEG2 in the BTR1 register, determines the number of clocks and sampling point position of each bit, with calculation formula as follows:

$$t_{SYNCSEG} = 1 * tscl$$

$$t_{TSEG1} = tscl * (8 * TSEG1.3 + 4 * TSEG1.2 + 2 * TSEG1.1 + TSEG1.0 + 1)$$

$$t_{TSEG2} = tscl * (4 * TSEG2.2 + 2 * TSEG2.1 + TSEG2.0 + 1)$$

20.1 Overview

The CAM (Camera) module is responsible for converting the parallel interface (DVP) into a general bus interface (AHB), and writing the pixel data generated by the image sensor into the system memory for subsequent image transmission or compression. The CAM module has a flexible output format configuration, which can meet a variety of image processing needs.

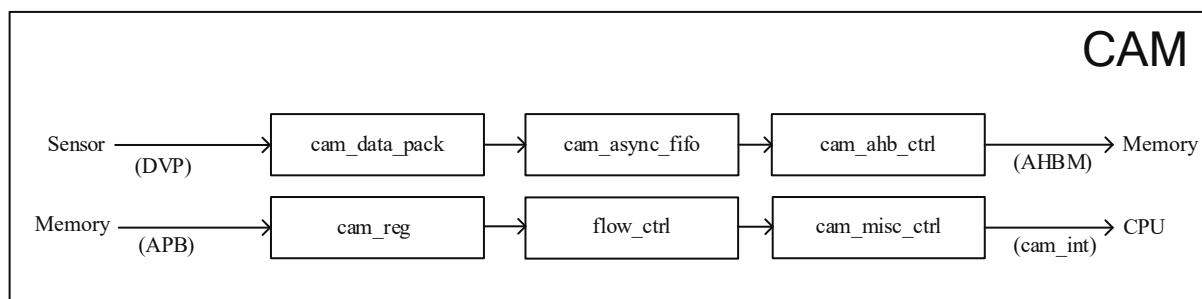


Fig. 20.1: CAM block diagram

20.2 Features

- Parallel interface 8-bit DVP signal, high-speed data transmission (80M), configurable DVP signal effective level and logic combination
- Support 8-bit/16-bit/24-bit input pixel width
- Support converting RGB888 input format to RGB565 or RGBA8888 output
- Support movie mode and photo mode
- Configurable drop patterns including:
 - Discard odd-digit data
 - Discard even-digit data

- Discard odd-numbered data in odd-numbered rows
- Discard even-digit data of odd-numbered rows
- Configurable image sensor line frame sync signal selection and polarity selection
- Support image rectangle cropping
- Frame selection function with a cycle of 1~32
- Support integrity detection of line frame synchronization signal
- AHB bus communication interface
- 512-byte buffer FIFO for occasional busy bus status
- Continuously cache up to 4 sets of image information
- A variety of application interruptions are conducive to flexible use and error prompts

20.3 Function description

20.3.1 DVP (Digital Video Port) signal and configuration

DVP (Digital Video Port) is a parallel interface, mainly including clock, frame synchronization, line synchronization and 8-bit data pins. The limit of the clock is limited to 80MHz, so it is generally used for sensors with resolutions below 5MP. The effective level of frame synchronization and line synchronization can be independently configured in the chip, and four modes are provided on the effective data:

- A. Frame sync and line sync are active at the same time (" & " logic)
- B. Either frame sync or line sync is valid ("| " logic)
- C. Frame synchronization is valid
- D. Line synchronization is valid

20.3.2 YCbCr format

The luminance signal is called Y, and the chrominance signal is composed of two independent signals. Depending on the color system and format, the two chrominance signals are often referred to as U, V or Pb, Pr or Cb, Cr. This results from different encoding formats, but in fact they have basically the same concept.

Since there are more retinal rod cells that recognize brightness than retinal cone cells that recognize chrominance on the human retina, the human eye is more sensitive to brightness than to chrominance. Therefore, part of the chrominance information can be discarded without being perceived by the human eye.

The format with the highest resolution of the chrominance signal is 4:4:4, that is, every 4-point Y sampling corresponds to 4-point Cb and 4-point Cr sampling. And 4:2:2 is that every 4 points of Y sampling corresponds to 2 points of Cb and 2 points of Cr sampling. In this format, the number of scan lines for chrominance signals is as many as that for

luminance signals, but the color of each scan line is The degree sample points are only half of the luminance signal. Different from the format mentioned above, 4:2:0 does not correspond to 2 points of Cb and 0 points of Cr sampling for every 4 points of Y sampling, but corresponds to 1 point of Cb and 1 point of Cr sampling for every 4 points of Y sampling. 4:0:0 is to discard all chrominance information, that is, the grayscale image.

20.3.3 Movie Mode/Photo Mode

In the photo mode, when the fixed-size memory given by the software is full, the CAM will stop, and the software will need to perform a POP operation to free up the space before continuing to write.

In video mode, it will keep rewriting on the fixed-size memory provided by the software, that is, using the memory as a ring buffer, without software for POP operation, to ensure that the image is taken out in real time or linked with the MJPEG module.

20.3.4 RGB888 to RGB565/RGBA8888 output

For image data whose input format is RGB888, you can choose to convert it to RGB565 or RGBA8888 and write it to the memory. If it is converted to RGB565 format, the arrangement order of R, G, B can be controlled by the bit FORMAT_565 of the register MISC. The arrangement order corresponding to different values is as follows:

- 0: byte2[7:3], byte1[7:2], byte0[7:3]
- 1: byte1[7:3], byte2[7:2], byte0[7:3]
- 2: byte2[7:3], byte0[7:2], byte1[7:3]
- 3: byte0[7:3], byte2[7:2], byte1[7:3]
- 4: byte1[7:3], byte0[7:2], byte2[7:3]
- 5: byte0[7:3], byte1[7:2], byte2[7:3]

If it is converted to RGBA8888 format, the value of A is the same as the value filled in bit ALPHA of the register MISC.

20.3.5 Image rectangle crop

By setting the start and end positions of the line synchronization signal and the frame synchronization signal cropping by the high 16 bits and the low 16 bits of the registers HSYNC_CONTROL and VSYNC_CONTROL, the image in the rectangular window of the specified position and size can be cropped, and the image beyond the rectangular part can be cropped. Data will be discarded. The start and end of the line synchronization signal are set to the pixel serial number, the start and end of the frame synchronization signal are set to the line number, and the cropped image contains the start point but not the end point.

20.3.6 Frame rounding function

A frame period n can be set through the register FRAME_PERIOD, the value range of n is 0~31, and the corresponding actual value is n+1, and then the register FRAME_VLD is used to set which frames of images are retained in a frame period. Write 1 in the corresponding bit position if you need to keep it, and write 0 in the corresponding bit position if you need to discard it. For example, if n is set to 5 and the value of FRAME_VLD is set to 0x13, in every 6 frames of images, the 1st, 2nd, and 5th frames will be written into the memory, and the 3rd, 4th, and 6th frames will be discarded. The cycle is performed with 6 frames of images as a cycle.

20.3.7 Line Frame Sync Signal Integrity Detection

The line synchronization signal comparison value and the frame synchronization signal comparison value can be set respectively through the lower 16 bits and the upper 16 bits of the register FRAME_SIZE_CONTROL, and the integrity of the signal can be detected. The line sync signal sets the total number of pixels in each row, and the frame sync signal sets the total number of lines. When the count value of the line or frame synchronization signal of a frame image is not equal to the comparison value, a corresponding interrupt will be generated.

20.3.8 Cache image information

The module contains 4 groups of FIFO to record the image address and image ID. Whenever this module completely writes a frame to the memory, it will record the start address and image ID of the frame image in this FIFO, but it should be noted that when the memory is insufficient, or the 4 sets of FIFO are full. In this case, the module will automatically discard the information of the next image. In the part where the image information is taken out, the oldest image information can be emptied by the pop operation. At this time, the FIFO will automatically advance to ensure the timing of the image information in the FIFO, as shown in the figure below:

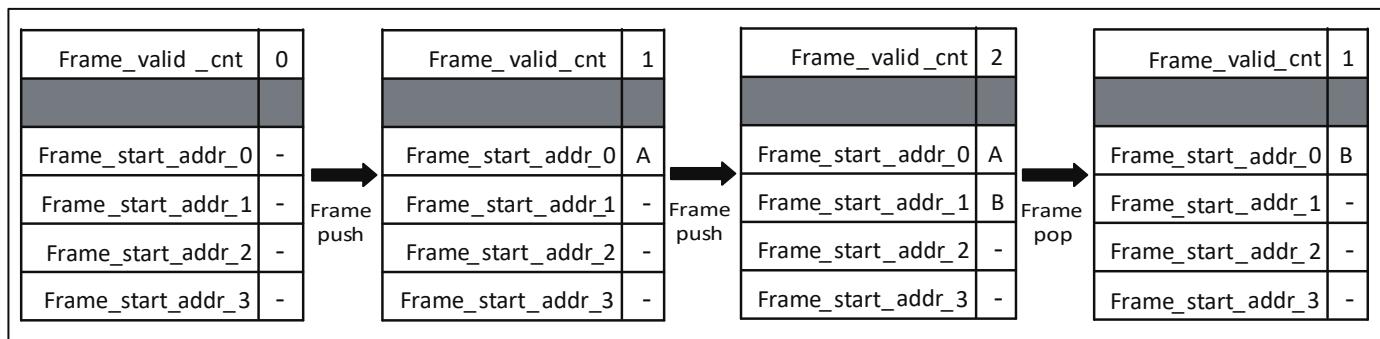


Fig. 20.2: FIFO framework

20.3.9 Support a variety of interrupt information (can be configured independently of the switch)

- A. Normal interrupt - a count value n can be set, and an interrupt will be issued every time n images are written
- B. Memory Interrupt - When the remaining memory space is less than one frame size, and the used memory is overwritten, an interrupt is issued, as shown in the following figure:

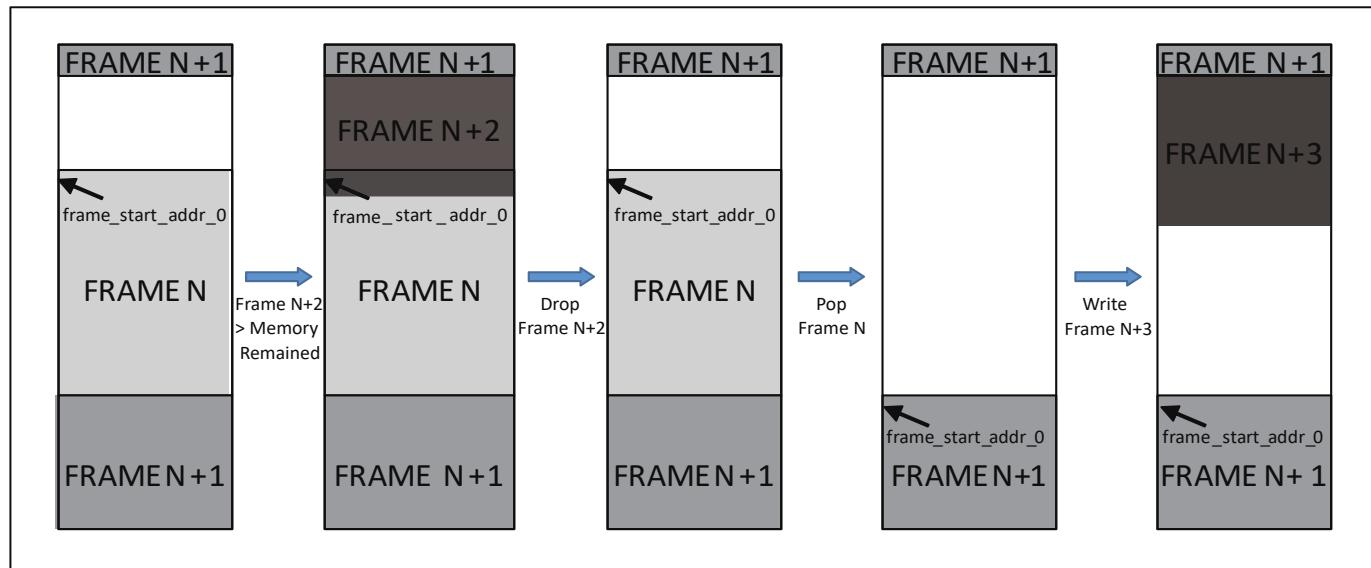


Fig. 20.3: Memory

- C. Frame interrupt - when there are more than 4 groups of unprocessed images and no more image information can be stored, an interrupt is issued
- D. FIFO interrupt - when the bus is too late to write to the memory, causing the FIFO to overflow, an interrupt is issued
- E. Hsync interrupt - when the number of pixels of a line in a frame of image is not equal to the set value (the line sync signal integrity test fails), an interrupt is issued
- F. Vsync interrupt - when the total number of lines in a frame of image is not equal to the set value (frame sync signal integrity check fails), an interrupt is issued

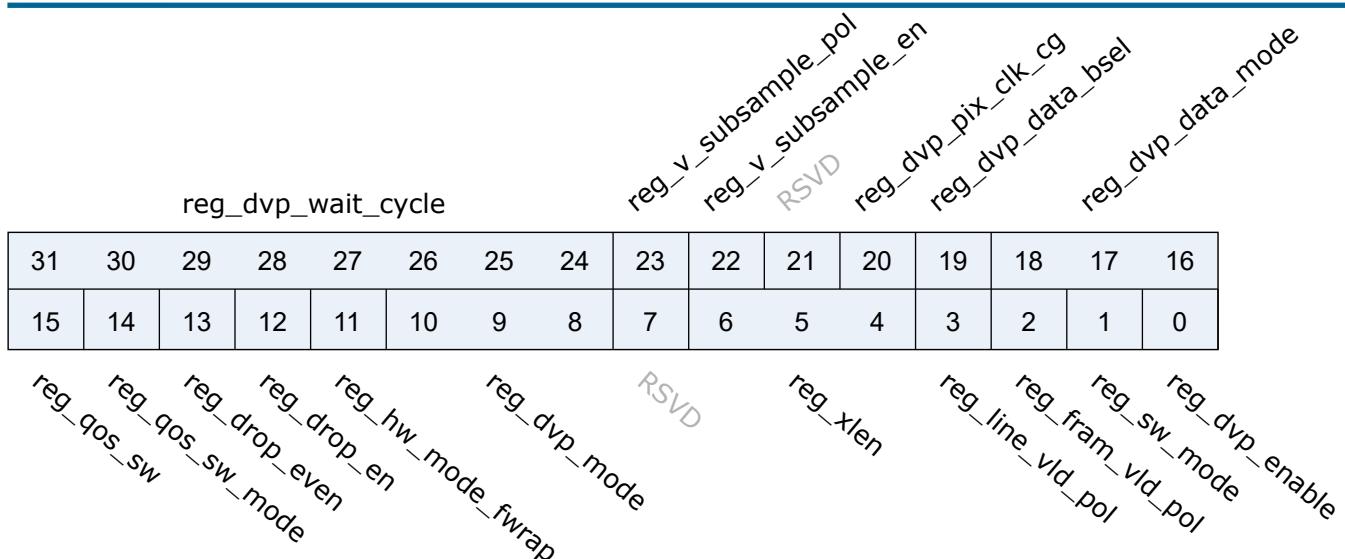
20.4 Register description

Name	Description
dvp2axi_configue	
dvp2axi_addr_start	
dvp2axi_mem_bcnt	
dvp_status_and_error	

Name	Description
dvp2axi_frame_bcnt	
dvp_frame_fifo_pop	
dvp2axi_frame_vld	
dvp2axi_frame_period	
dvp2axi_misc	
dvp2axi_hsync_crop	
dvp2axi_vsync_crop	
dvp2axi_fram_exm	
frame_start_addr0	
frame_start_addr1	
frame_start_addr2	
frame_start_addr3	
frame_id_sts01	
frame_id_sts23	
dvp_debug	
dvp_dummy_reg	

20.4.1 dvp2axi_configue

Address: 0x30015000



Bits	Name	Type	Reset	Description
31:24	<i>reg_dvp_wait_cycle</i>	r/w	8'h40	Cycles in FSM Wait mode
23	<i>reg_v_subsample_pol</i>	r/w	1'b0	DVP2BUS vertical sub-sampling polarity 1'b0: Odd lines are masked 1'b1: Even lines are masked
22	<i>reg_v_subsample_en</i>	r/w	1'b0	DVP2BUS vertical sub-sampling enable
21	RSVD			
20	<i>reg_dvp_pix_clk_cg</i>	r/w	1'b0	DVP pix clk gate
19	<i>reg_dvp_data_bsel</i>	r/w	1'b0	Byte select signal for DVP 8-bit mode, don't care if <i>reg_dvp_data_8bit</i> is disabled 1'b0: Select the lower byte of <i>pix_data</i> 1'b1: Select the upper byte of <i>pix_data</i>
18:16	<i>reg_dvp_data_mode</i>	r/w	3'b0	DVP 8-bit mode enable 3'd0: DVP <i>pix_data</i> is 16-bit wide 3'd1: DVP <i>pix_data</i> is 24-bit mode 3'd2: DVP <i>pix_data</i> is 24-comp-16-bit mode 3'd3: DVP <i>pix_data</i> is 24-exp-32-bit mode 3'd4: DVP <i>pix_data</i> is 8-bit wide Others - Reserved
15	<i>reg_qos_sw</i>	r/w	1'b0	AXI Qos software mode value
14	<i>reg_qos_sw_mode</i>	r/w	1'b0	AXI QoS software mode enable
13	<i>reg_drop_even</i>	r/w	1'b0	Only effect when <i>reg_drop_en</i> =1 : 1'b1 : Drop all even bytes 1'b0 : Drop all odd bytes
12	<i>reg_drop_en</i>	r/w	1'b0	Drop mode Enable

Bits	Name	Type	Reset	Description
11	reg_hw_mode_fwrap	r/w	1'b1	DVP2BUS HW mode with frame start address wrap to reg_addr_start
10:8	reg_dvp_mode	r/w	3'd0	Image sensor mode selection: 3'd0-Vsync&Hsync 3'd1-Vsync Hsync 3'd2-Vsync 3'd3-Hsync
7	RSVD			
6:4	reg_xlen	r/w	3'd3	burst length setting 3'd0 - Single / 3'd1 - INCR4 / 3'd2 - INCR8 3'd3 - INCR16 / 3'd5 - INCR32 / 3'd6 - INCR64
3	reg_line_vld_pol	r/w	1'b1	Image sensor line valid polarity, 1'b0 - Active low, 1'b1 - Active high
2	reg_fram_vld_pol	r/w	1'b1	Image sensor frame valid polarity, 1'b0 - Active low, 1'b1 - Active high
1	reg_sw_mode	r/w	1'b0	DVP2BUS SW manual mode (don't care if reg_swap_mode is enabled)
0	reg_dvp_enable	r/w	1'b0	module enable

20.4.2 dvp2axi_addr_start

Address: 0x30015004

reg_addr_start

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

reg_addr_start

Bits	Name	Type	Reset	Description
31:0	reg_addr_start	r/w	32'h80000000	AXI start address

20.4.3 dvp2axi_mem_bcnt

Address: 0x30015008

reg_mem_burst_cnt

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

reg_mem_burst_cnt

Bits	Name	Type	Reset	Description
31:0	reg_mem_burst_cnt	r/w	32'hC000	AXI burst cnt before wrap to "reg_addr_start"

20.4.4 dvp_status_and_error

Address: 0x3001500c

RSVD	RSVD	st_dvp_idle	axi_idle	st_bus_fls	st_bus_wait	st_bus_func	st_bus_idle	RSVD	sts_vcint_int	sts_hcint_int	frame_valid_cnt				
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

sts_fifo_int	sts_frame_int	sts_mem_int	sts_normal_int	reg_int_fifo_en	reg_int_frame_en	reg_int_mem_en	reg_int_normal_en	reg_int_vcint_en	reg_int_hcint_en	RSVD	reg_frame_cnt_trgr_int				

Bits	Name	Type	Reset	Description
31:30	RSVD			
29	st_dvp_idle	r	1'b1	DVP2BUS asynchronous fifo idle status
28	axi_idle	r	1'b1	DVP2BUS AHB idle status
27	st_bus_fls	r	1'b0	DVP in flush state
26	st_bus_wait	r	1'b0	DVP in wait state
25	st_bus_func	r	1'b0	DVP in functional state
24	st_bus_idle	r	1'b1	DVP in idle state
23	RSVD			
22	sts_vcint_int	r	1'b0	Vsync valid line count non-match interrupt status

Bits	Name	Type	Reset	Description
21	sts_hcnt_int	r	1'b0	Hsync valid pixel count non-match interrupt status
20:16	frame_valid_cnt	r	5'd0	Frame counts in memory before read out in SW mode
15	sts_fifo_int	r	1'b0	FIFO OverWrite interrupt status
14	sts_frame_int	r	1'b0	Frame OverWrite interrupt status
13	sts_mem_int	r	1'b0	Memory OverWrite interrupt status
12	sts_normal_int	r	1'b0	Normal Write interrupt status
11	reg_int_fifo_en	r/w	1'b1	FIFO OverWrite interrupt enable
10	reg_int_frame_en	r/w	1'b0	Frame OverWrite interrupt enable
9	reg_int_mem_en	r/w	1'b0	Memory OverWrite interrupt enable
8	reg_int_normal_en	r/w	1'b0	Normal Write interrupt enable
7	reg_int_vcnt_en	r/w	1'b0	Vsync valid line count match interrupt enable
6	reg_int_hcnt_en	r/w	1'b0	Hsync valid pixel count match interrupt enable
5	RSVD			
4:0	reg_frame_cnt_trgr_int	r/w	5'd0	Frame to issue interrupt at SW Mode

20.4.5 dvp2axi_frame_bcnt

Address: 0x30015010

reg_frame_byte_cnt

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

reg_frame_byte_cnt

Bits	Name	Type	Reset	Description
31:0	reg_frame_byte_cnt	r/w	32'h7e90	Single Frame byte cnt(Need pre-calculation)

20.4.6 dvp_frame_fifo_pop

Address: 0x30015014

RSVD															
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
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	reg_int_vcnt_clr	reg_int_hcnt_clr	reg_int_fifo_clr	reg_int_frame_clr	reg_int_mem_clr	reg_int_normal_clr	RSVD	RSVD	RSVD	r fifo_pop

Bits	Name	Type	Reset	Description
31:10	RSVD			
9	reg_int_vcnt_clr	w1p	1'd0	Interrupt clear
8	reg_int_hcnt_clr	w1p	1'd0	Interrupt clear
7	reg_int_fifo_clr	w1p	1'd0	Interrupt clear
6	reg_int_frame_clr	w1p	1'd0	Interrupt clear
5	reg_int_mem_clr	w1p	1'd0	Interrupt clear
4	reg_int_normal_clr	w1p	1'd0	Interrupt clear
3:1	RSVD			
0	r fifo_pop	w1p	1'b0	Write this bit will trigger fifo pop

20.4.7 dvp2axi_frame_vld

Address: 0x30015018

reg_frame_n_vld															
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

Bits	Name	Type	Reset	Description
31:0	reg_frame_n_vld	r/w	32'hffff_- ffff	Bitwise frame valid in period

20.4.8 dvp2axi_frame_period

Address: 0x3001501c

RSVD															
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

reg_frame_period

Bits	Name	Type	Reset	Description
31:5	RSVD			
4:0	reg_frame_period	r/w	5'h0	Frame period cnt. (EX. Set this register 0, the period is 1)

20.4.9 dvp2axi_misc

Address: 0x30015020

RSVD															
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

reg_alpha

reg_format_565

Bits	Name	Type	Reset	Description
31:11	RSVD			
10:8	reg_format_565	r/w	3'd0	Only work when reg_dvp_data_mode=2 (24-comp-16-bit mode) 3'd0: B2(5)B1(6)B0(5) 3'd1: B1(5)B2(6)B0(5) 3'd2: B2(5)B0(6)B1(5) 3'd3: B0(5)B2(6)B1(5) 3'd4: B1(5)B0(6)B2(5) 3'd5: B0(5)B1(6)B2(5)

Bits	Name	Type	Reset	Description
7:0	reg_alpha	r/w	8'h0	Only work when "reg_dvp_data_mode==2'd3(DVP pix_data is 24-exp-32-bit mode)" The value of [31:24]

20.4.10 dvp2axi_hsync_crop

Address: 0x30015030

reg_hsync_act_start

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

reg_hsync_act_end

Bits	Name	Type	Reset	Description
31:16	reg_hsync_act_start	r/w	16'h0	Valid hsync start cnt
15:0	reg_hsync_act_end	r/w	16'hFFFF	Valid hsync end cnt

20.4.11 dvp2axi_vsync_crop

Address: 0x30015034

reg_vsync_act_start

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

reg_vsync_act_end

Bits	Name	Type	Reset	Description
31:16	reg_vsync_act_start	r/w	16'h0	Valid vsync start cnt
15:0	reg_vsync_act_end	r/w	16'hFFFF	Valid vsync end cnt

20.4.12 dvp2axi_fram_exm

Address: 0x30015038

reg_total_vcnt

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

reg_total_hcnt

Bits	Name	Type	Reset	Description
31:16	reg_total_vcnt	r/w	16'h0	Total valid line count in a frame
15:0	reg_total_hcnt	r/w	16'h0	Total valid pix count in a line

20.4.13 frame_start_addr0

Address: 0x30015040

frame_start_addr_0

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

frame_start_addr_0

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_0	r	32'd0	DVP2BUS PIC 0 Start address

20.4.14 frame_start_addr1

Address: 0x30015048

frame_start_addr_1

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

frame_start_addr_1

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_1	r	32'd0	DVP2BUS PIC 1 Start address

20.4.15 frame_start_addr2

Address: 0x30015050

frame_start_addr_2

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

frame_start_addr_2

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_2	r	32'd0	DVP2BUS PIC 2 Start address

20.4.16 frame_start_addr3

Address: 0x30015058

frame_start_addr_3

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

frame_start_addr_3

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_3	r	32'd0	DVP2BUS PIC 3 Start address

20.4.17 frame_id_sts01

Address: 0x30015060

frame_id_1

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

frame_id_0

Bits	Name	Type	Reset	Description
31:16	frame_id_1	r	16'd0	DVP2BUS PIC 1 ID
15:0	frame_id_0	r	16'd0	DVP2BUS PIC 0 ID

20.4.18 frame_id_sts23

Address: 0x30015064

frame_id_3

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

frame_id_2

Bits	Name	Type	Reset	Description
31:16	frame_id_3	r	16'd0	DVP2BUS PIC 3 ID
15:0	frame_id_2	r	16'd0	DVP2BUS PIC 2 ID

20.4.19 dvp_debug

Address: 0x300150f0

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
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	
RSVD	RSVD	RSVD	RSVD	reg_id_latch_line				RSVD	RSVD	RSVD	RSVD	reg_dvp_dbg_sel				reg_dvp_dbg_en

Bits	Name	Type	Reset	Description
31:12	RSVD			
11:8	reg_id_latch_line	r/w	4'd5	ID latch timing (line count)
7:4	RSVD			
3:1	reg_dvp_dbg_sel	r/w	3'd0	DVP2BUS debug flag selection
0	reg_dvp_dbg_en	r/w	1'b0	DVP2BUS debug flag enable

20.4.20 dvp_dummy_reg

Address: 0x300150fc

RESERVED

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

RESERVED

Bits	Name	Type	Reset	Description
31:0	RESERVED	rsvd	32'hf0f0f0f0	RESERVED

21.1 Overview

MJPEG (Motion Joint Photographic Experts Group) is a video coding format that can be accurate to frame editing and multi-layer image processing. It handles motion video sequences as continuous still images. This compression method compresses each frame individually and completely. . By compressing the original data in YCbCr format, the memory space occupied by one frame of image can be greatly reduced.

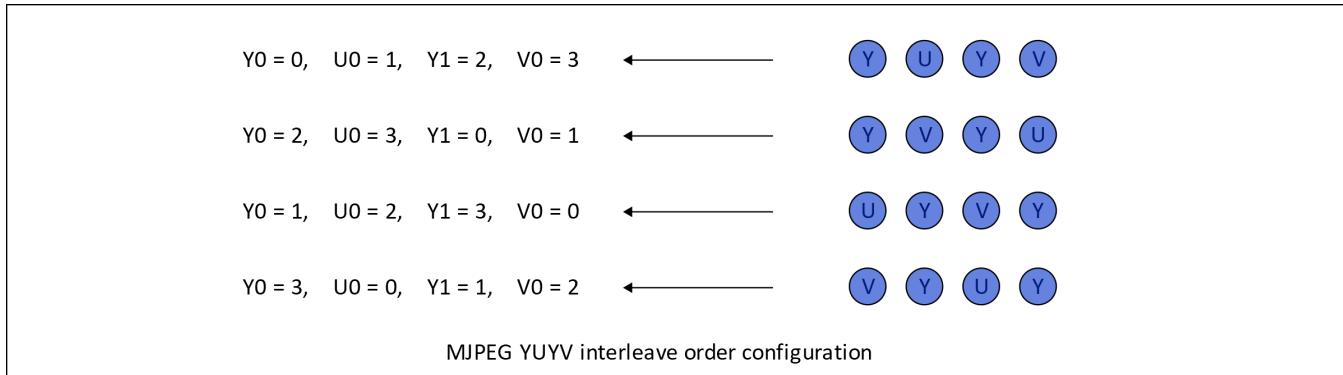
21.2 Features

- Configurable input formats including:
 - YCbCr4:2:2 plane or packed mode
 - YCbCr4:2:0 plane mode
 - YCbCr4:0:0
- Configurable input data Y, Cb, Cr sort order
- Quantization coefficient table can be freely configured
- Support software mode and linkage mode
- Support swap mode
- Support kick mode
- Reserve jpg head space and automatically add jpg tail
- Continuously cache up to 4 sets of picture information
- A variety of application interruptions are conducive to flexible use and error prompts

21.3 Function description

21.3.1 Input configuration

The YCbCr format of the input data can be selected by bit <REG_YUV_MODE> of register MJPEG_CONTROL_1, including YCbCr4:2:2 plane or packed mode, YCbCr4:2:0 plane mode and YCbCr4:0:0 grayscale image. When the YCbCr4:2:2 packing mode is selected, the arrangement order of Y, Cb and Cr can be configured through the upper 8 bits of the register MJPEG_HEADER_BYTE. When other modes are selected, the order of Cb and Cr can be set by bit <REG_ORDER_U_EVEN> of register MJPEG_CONTROL_1. The detailed configuration is shown in the figure below.



21.3.2 Quantization coefficient table

The quantization coefficient table can be freely configured by the user, <reg_q_0_00> to <reg_q_0_3F> represent the quantization tables of the Y component of grayscale information, and <reg_q_1_00> to <reg_q_1_3F> represent the quantization tables of the chrominance information Cb and Cr. The quantization table is arranged in the order from the upper left corner to the bottom, the first column is followed by the second column, that is, reg_q_0_00 represents the quantization value of the first row and the first column of the Y component, and reg_q_0_01 represents the quantization of the second row and the first column of the Y component. Value, reg_q_0_07 represents the quantized value of the Y component in the eighth row and first column, reg_q_0_08 represents the quantized value in the first row and second column of the Y component, and so on.

21.3.3 Software mode and link mode

When the bit <REG_MJPEG_SW_MODE> of the MJPEG_CONTROL_2 register is set to 1, the software mode is enabled. In this mode, MJPEG will read the original image data of the specified number of frames from the memory for compression. In this mode, the image data needs to be prepared in advance.

When the bit <REG_MJPEG_SW_MODE> of the MJPEG_CONTROL_2 register is cleared to 0, the linkage mode is enabled. In this mode, MJPEG will process the output of the CAM module as its input. MJPEG is processed with 8*8 data blocks as a unit. When the CAM finishes writing 8 lines of data to the memory, MJPEG will start to work, and the memory space processed by MJPEG will be released to the CAM for reuse, so that the CAM can complete the linkage with MJPEG without the memory space of a whole picture.

21.3.4 Swap mode

When using swap mode, the MJPEG storage space will be evenly divided into two blocks, when MJPEG finishes writing one block, an interrupt will be generated to notify the software to read the data, and it will write the data to the other block. Alternate use back and forth, so as to use less than one frame of picture storage space for data processing.

21.3.5 Kick mode

The kick mode is enabled when bit <REG_SW_KICK_MODE> of the MJPEG_CONTROL_2 register is set to 1. In this mode, each time a 1 is written to the bit <REG_SW_KICK> of the MJPEG_CONTROL_2 register, a compression is started. The number of compressed lines is determined by the bit <REG_SW_KICK_HBLK> of the MJPEG_YUV-MEM_SW register. It should be noted that the kick mode can only be used in software mode, not in linkage mode.

21.3.6 Jpg function

The jpg function will automatically reserve a certain number of bytes of space at the beginning of each frame of data and fill it with the specified data. The size of this space is set by the lower 12 bits of the register MJPEG_HEADER-BYTE. There is 768 bytes of memory at offset 0x800 for storing the data that needs to be filled at the beginning of each frame. The user only needs to write the jpg header information into it, and this information will be included at the beginning of each frame of data generated. In addition, if the auto-fill jpg end is enabled, two bytes of data will be added at the end of each frame of data, and the value of these two bytes is 0xFFD9.

21.3.7 Cache image information

The module contains 4 groups of FIFO to record the picture address, picture size and picture ID. Whenever this module completely writes a frame to the memory, it will record the start address, image size and image ID of the frame image in this FIFO, but it should be noted that when the memory is insufficient, or there are 4 sets of FIFO. When the storage is full, the module will automatically discard the information of the next picture. In the part where the picture information is taken out, the pop action can be performed through the APB interface to empty the oldest picture information. At this time, the FIFO will be automatically advanced to ensure the FIFO. Timing of internal picture information.

21.3.8 Support a variety of interrupt information (can be configured independently of the switch)

- A. Normal interrupt - can be set to interrupt after writing several pictures
- B. Camera Interrupt - When the speed of MJPEG processing cannot keep up with the speed of CAM module writing, and the CAM overflows, an interrupt is issued
- C. Memory interrupt - when the memory is overwritten, an interrupt is issued
- D. Frame interrupt - when there are more than 4 groups of unprocessed pictures, an interrupt is issued

E. Swap interrupt - when a memory block is full in swap mode, an interrupt is issued

22.1 Overview

The module also integrates the Quad Serial Peripheral Interface (QSPI) display interface. QSPI with four data lines is an extension of the SPI interface and it greatly improves the transmission efficiency.

22.2 Features

- Additional support for QSPI mode, with the following features:
 - * A single transmission includes a one-byte command phase, an adjustable address phase of one to three bytes, and an optional data phase.
 - * Command, address and data reading and writing all support 1-wire /4-wire mode, which can be combined as needed.
- Except for QSPI mode, a single transmission supports optional command and data phases, and the QSPI interface has another address phase.
- The data phase can be set to normal mode in bytes and pixel mode in pixels:
 - * The data phase (normal mode) is used to transmit configuration data. It can write up to 256 bytes and read up to 8 bytes at a time.
 - * The data phase (pixel mode) is used to transmit pixel data, and the data size is in pixels. It can write 2 to the power of 24 pixels at a time and is unreadable.
- The input pixel formats support RGB and YUV444, in which RGB supports eight memory arrangements:
- YUV444 supports six memory arrangements:
 - * YUVN8888
 - * VUYN8888
 - * NYUV8888
 - * NVUY8888
 - * YUV888
 - * VUY888
- Hardware transcoding from YUV444 to RGB565/666/888
- The CS signal pull-up release condition can be configured

22.3 Function description

The basic block diagram of the DBI module is shown in the figure.

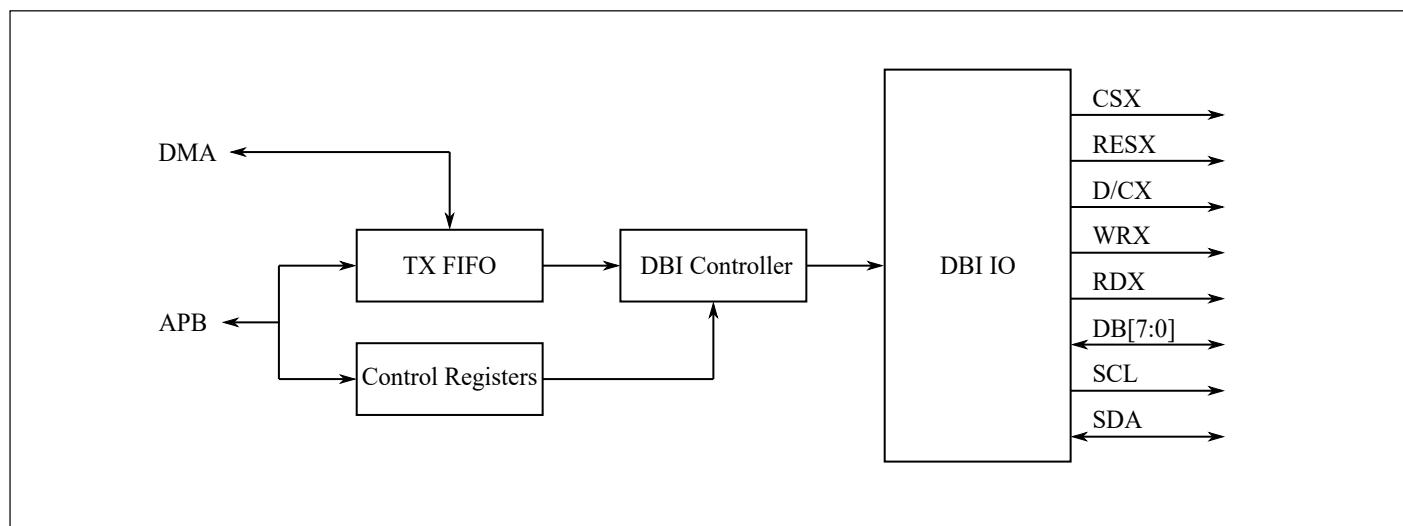


Fig. 22.1: DBI basic block diagram

22.3.1 DBI Type B

22.3.1.1 Write timing

The DBI Type B mode has an 8-bit parallel data line. The sequence of writing data from the MCU to the display module is shown in the following figure:

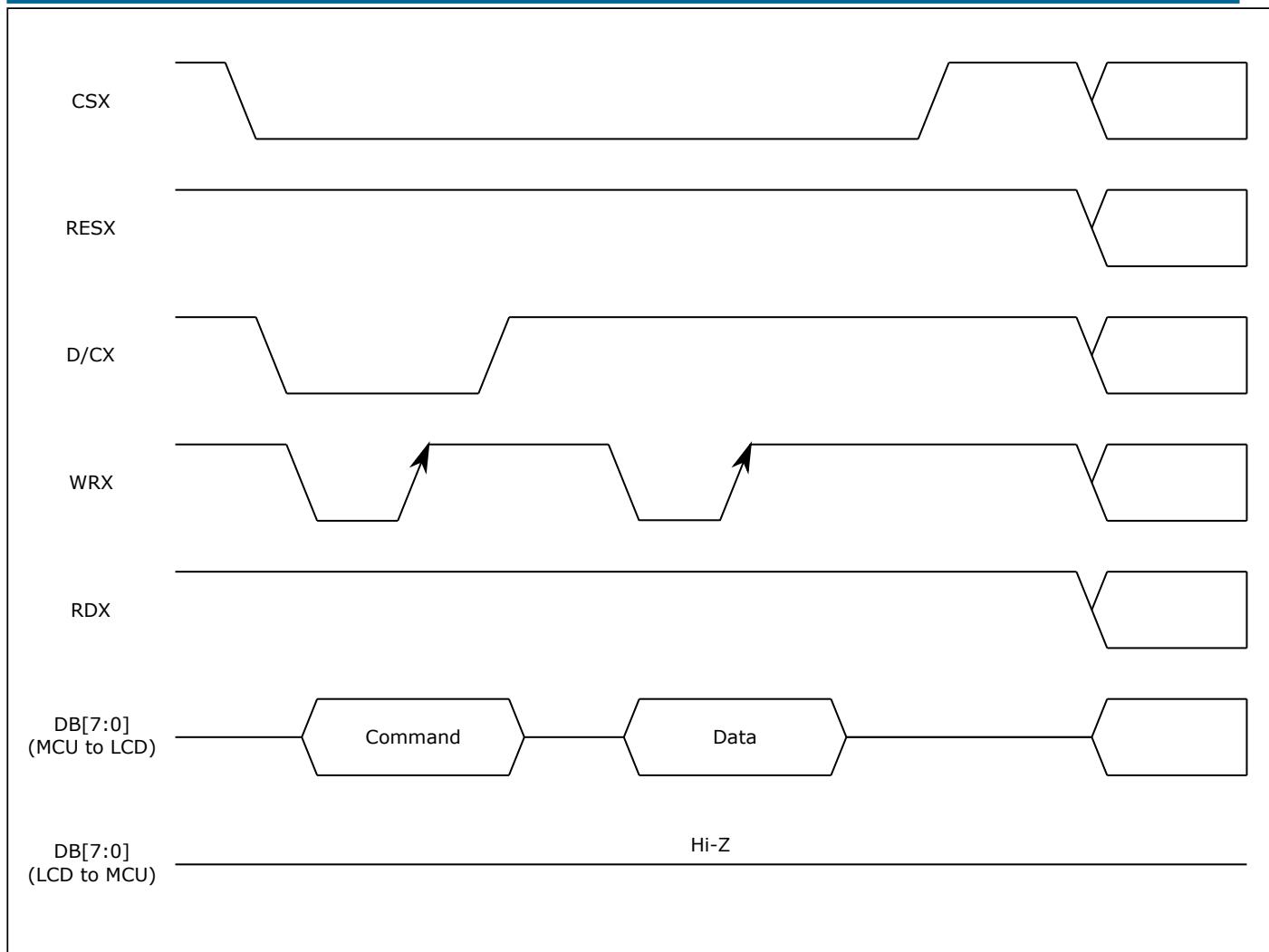


Fig. 22.2: Write timing

- CSX: Chip select signal. When the signal is low, the display module is selected, and when it is high, the display module will ignore all other interface signals
- RESX: External reset signal. When the signal is low, the display module is reset
- D/CX: Data/Command selection signal. When the signal is 0, the DB[7:0] bit is the command; when the signal is 1, the DB[7:0] bit is the RAM data or command parameter
- WRX: Parallel data write strobe signal. This signal is driven high to low during the write cycle and then pulled back high. The display module reads the information transmitted by the MCU at the rising edge of this signal. The signal is an asynchronous signal that can be terminated when not in use
- RDX: Parallel data read strobe signal. This signal is driven high to low and then pulled back high during the read cycle. The MCU reads the information transmitted by the display module at the rising edge of this signal. The signal is an asynchronous signal that can be terminated when not in use
- DB[7:0]: 8-bit data signal. Used to transfer commands, command parameters, or data

22.3.1.2 Read timing

The sequence of MCU reading data from the display module is shown in the following figure:

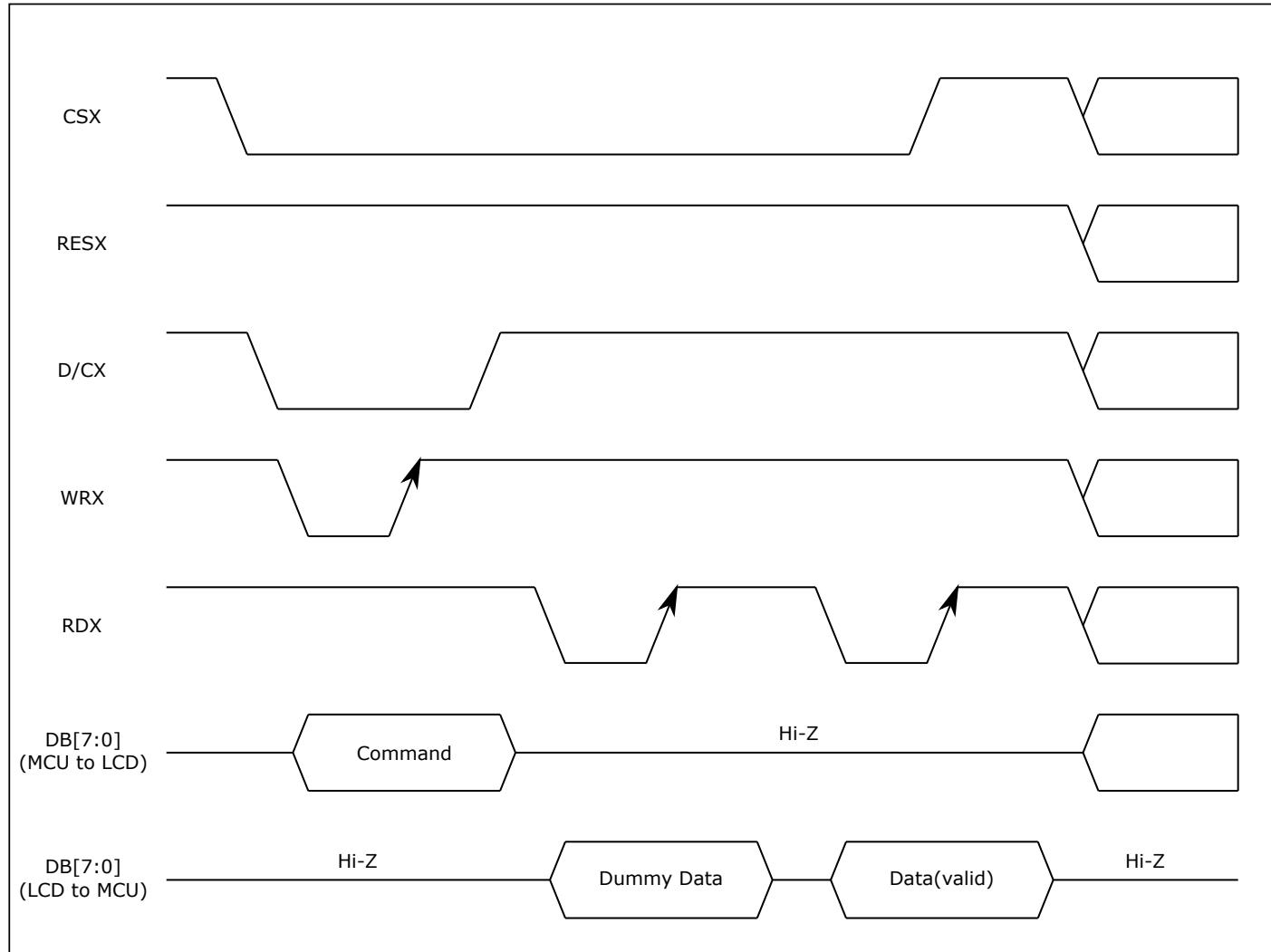


Fig. 22.3: Read timing

22.3.1.3 Output RGB565

The data output in RGB565 format is shown in the following figure:

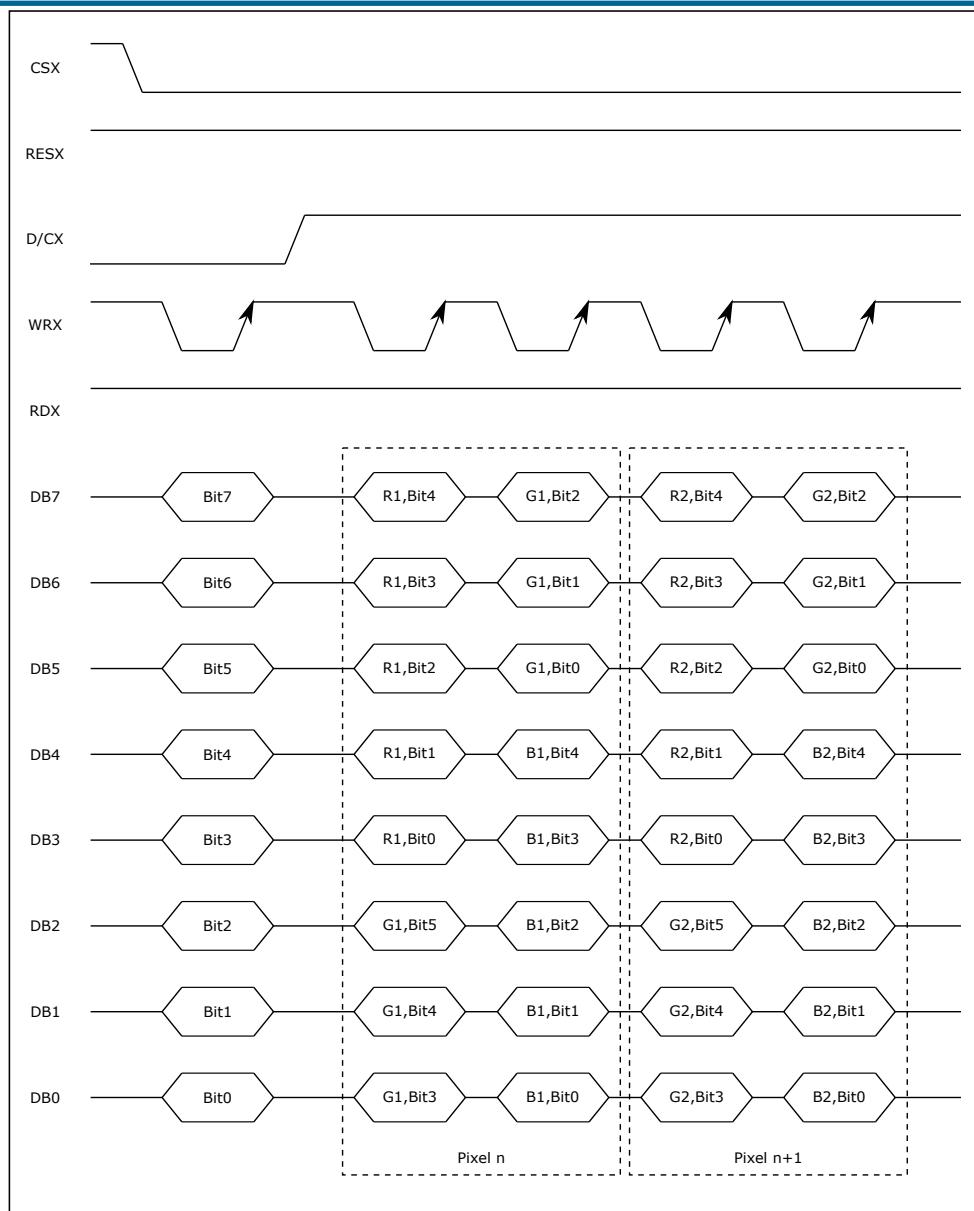


Fig. 22.4: RGB565 output

22.3.1.4 Output RGB666

The data output in RGB666 format is shown in the following figure:

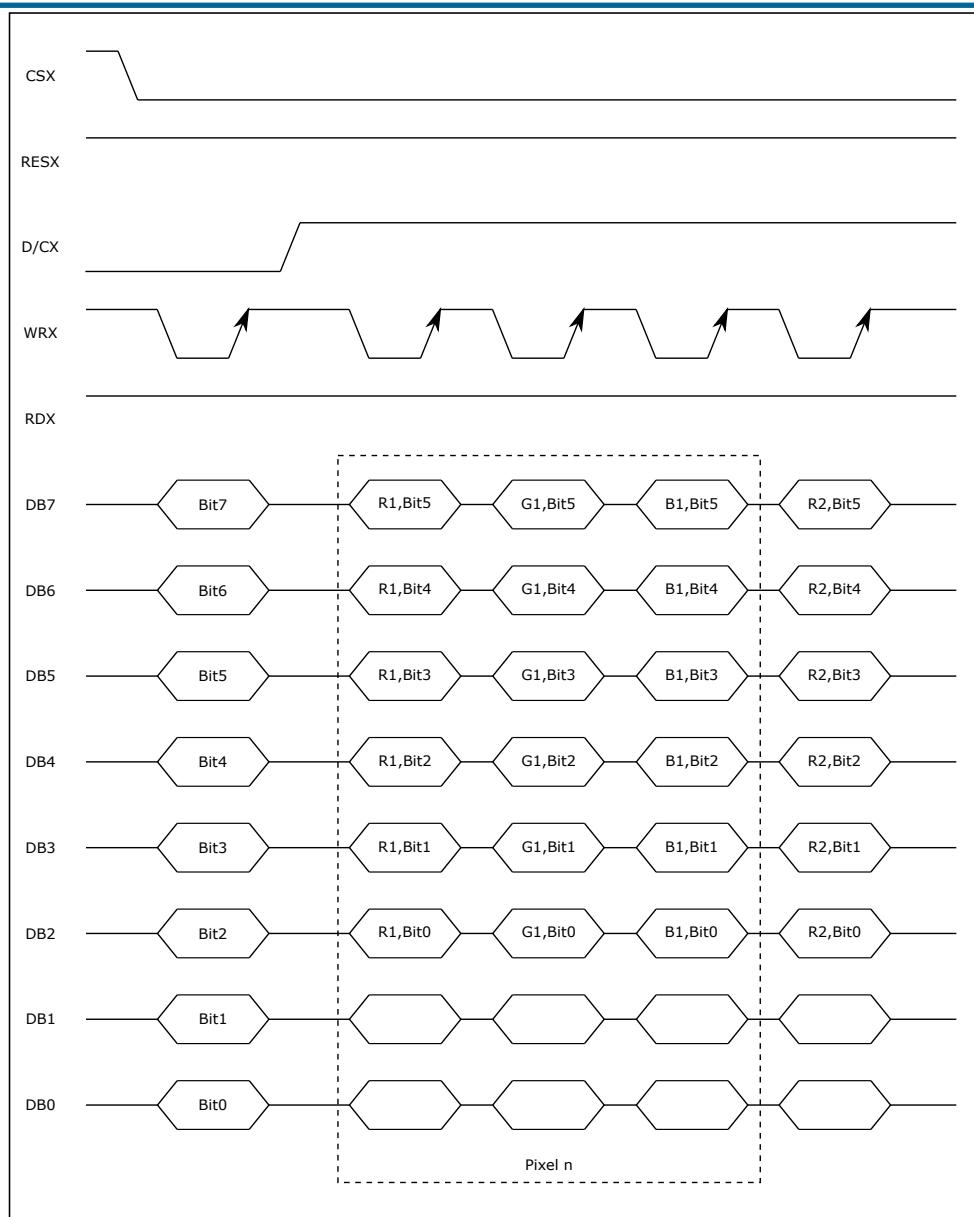


Fig. 22.5: RGB666 output

22.3.1.5 Output RGB888

The data output in RGB888 format is shown in the following figure:

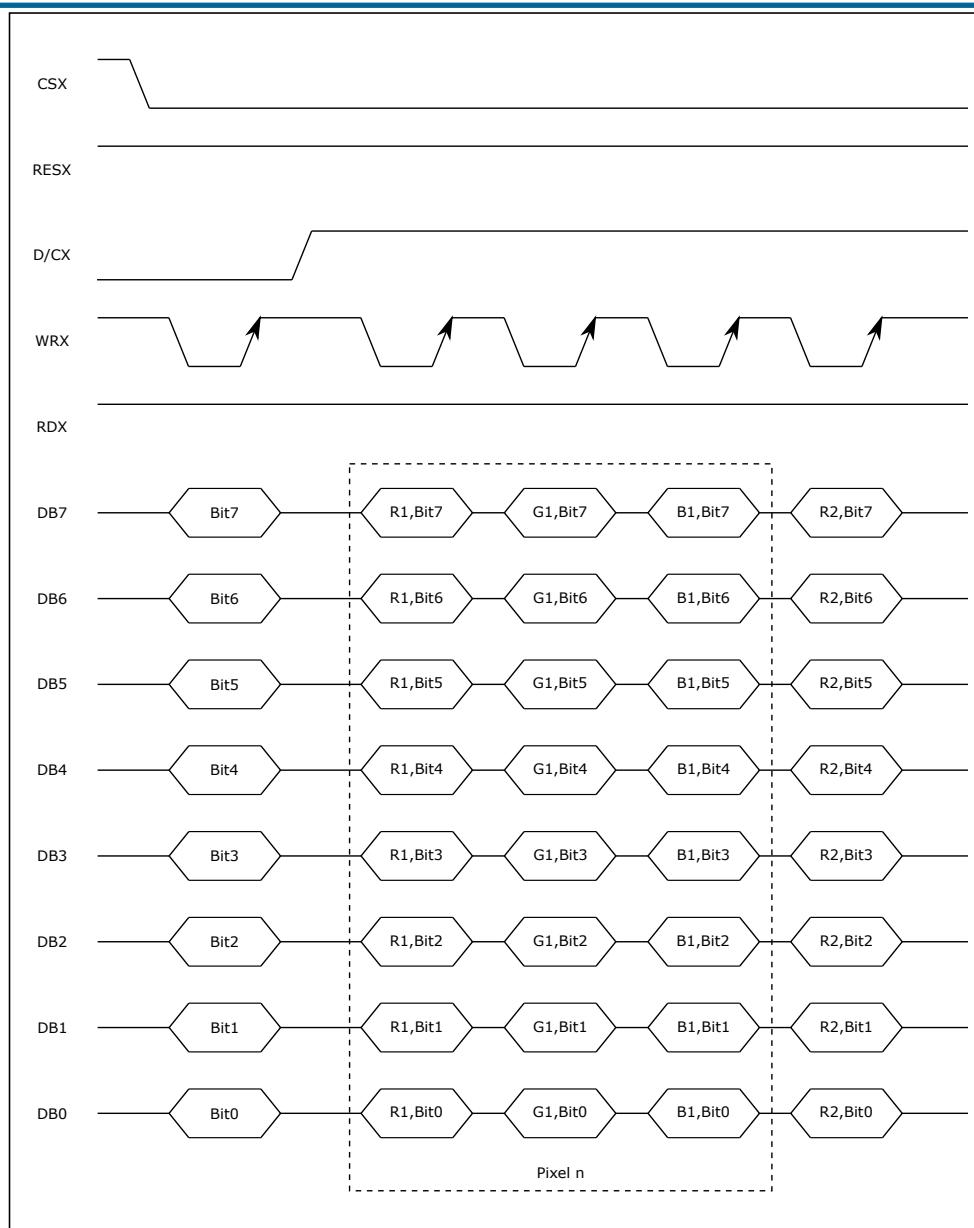


Fig. 22.6: RGB888 output

22.3.2 DBI Type C 3-Line

22.3.2.1 Write timing

The DBI Type C mode is a 3-wire 9-bit serial interface. The sequence of writing data from the MCU to the display module is shown in the following figure:

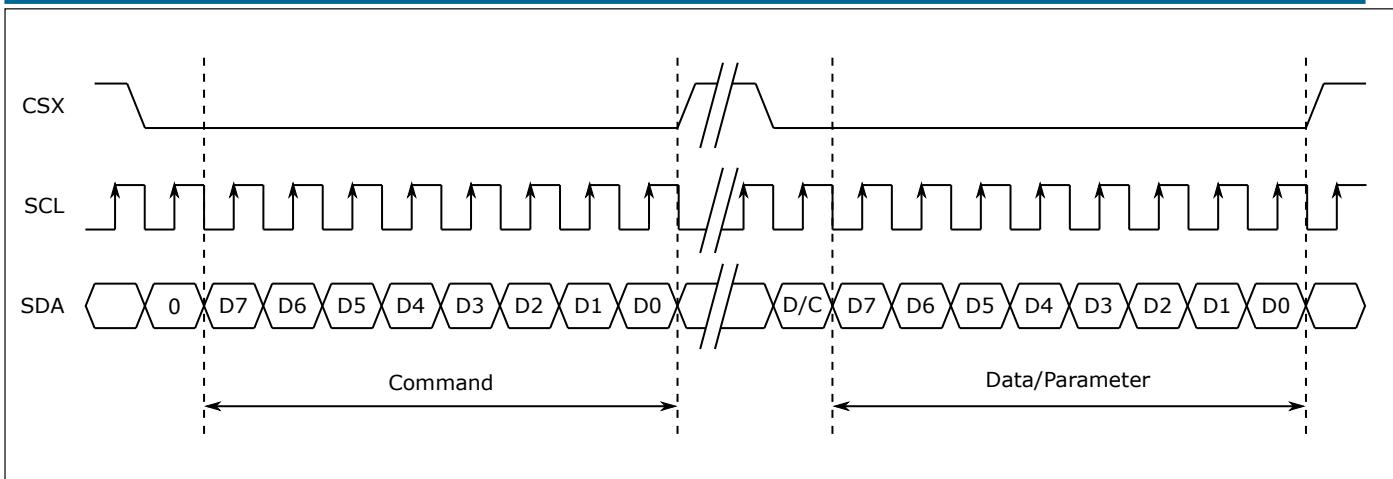


Fig. 22.7: Write timing

- CSX: Chip select signal. When the signal is low, the display module is selected, and when it is high, the display module will ignore all other interface signals;
- SCL: Serial clock input signal. Used to provide a clock signal for data transmission.
- SDA: Serial data input/output signal. In a write operation, the serial data packet contains a D/CX (Data/Command) select bit and a transfer byte. If the D/CX bit is low, the transmitted byte is a command; if the D/CX bit is high, the transmitted byte is display data or command parameters.

22.3.2.2 Read timing

The sequence of MCU reading data from the display module is shown in the following figure:

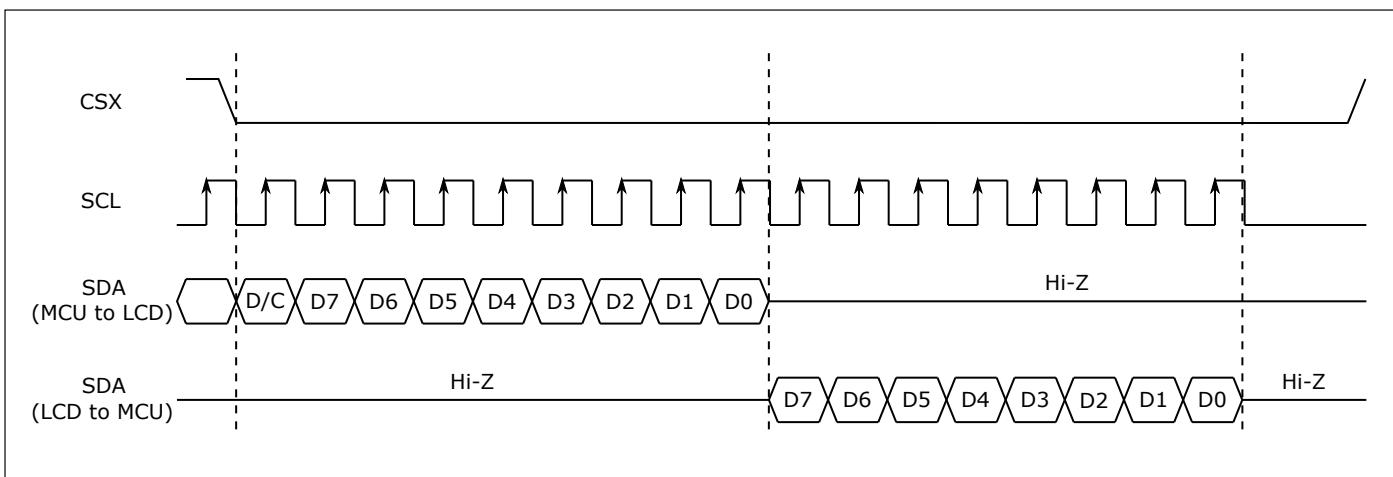


Fig. 22.8: Read timing

22.3.2.3 Output RGB565

The data output in RGB565 format is shown in the following figure:

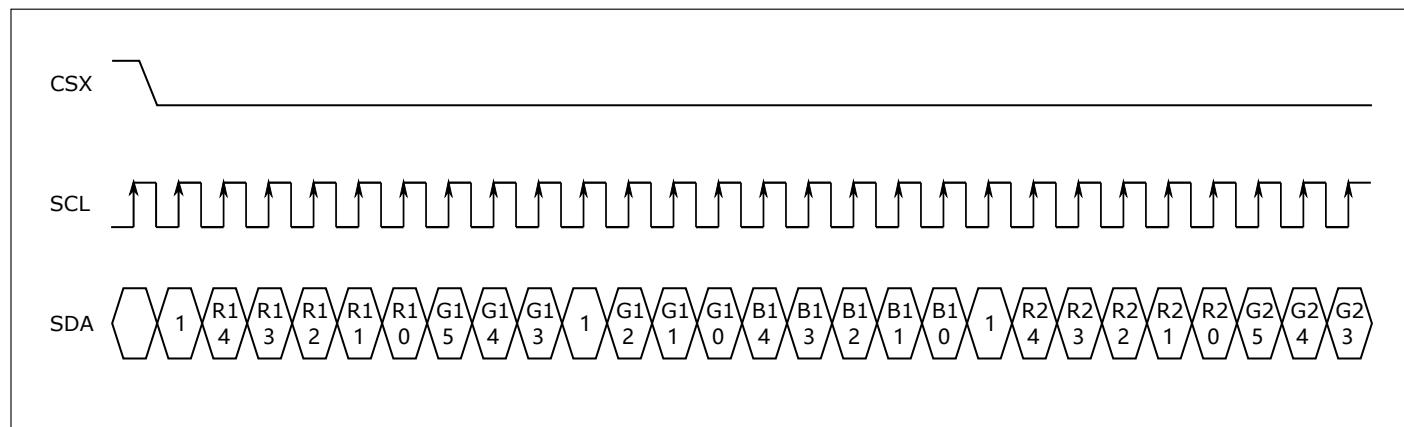


Fig. 22.9: RGB565 output

22.3.2.4 Output RGB666

The data output in RGB666 format is shown in the following figure:

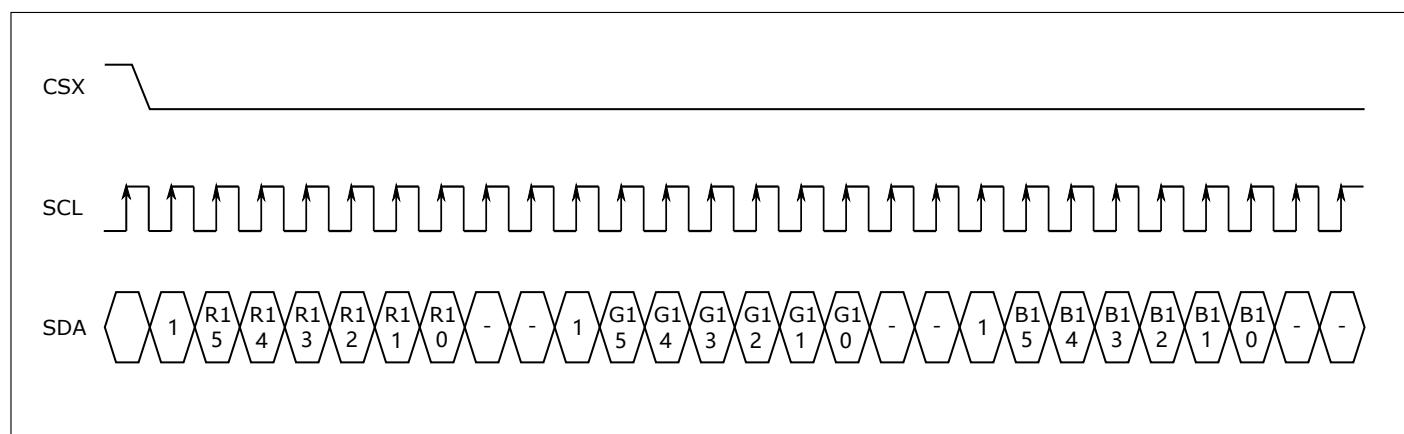


Fig. 22.10: RGB666 output

22.3.2.5 Output RGB888

The data output in RGB888 format is shown in the following figure:

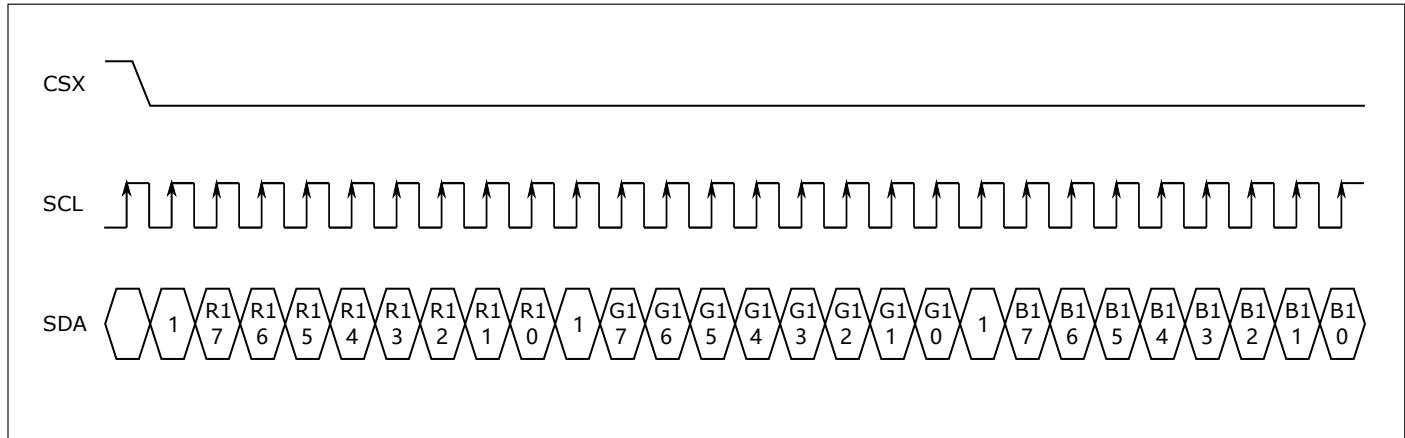


Fig. 22.11: RGB888 output

22.3.3 DBI Type C 4-Line

22.3.3.1 Write timing

The DBI Type C mode is a 4-wire 8-bit serial interface. The sequence of writing data from the MCU to the display module is shown in the following figure:

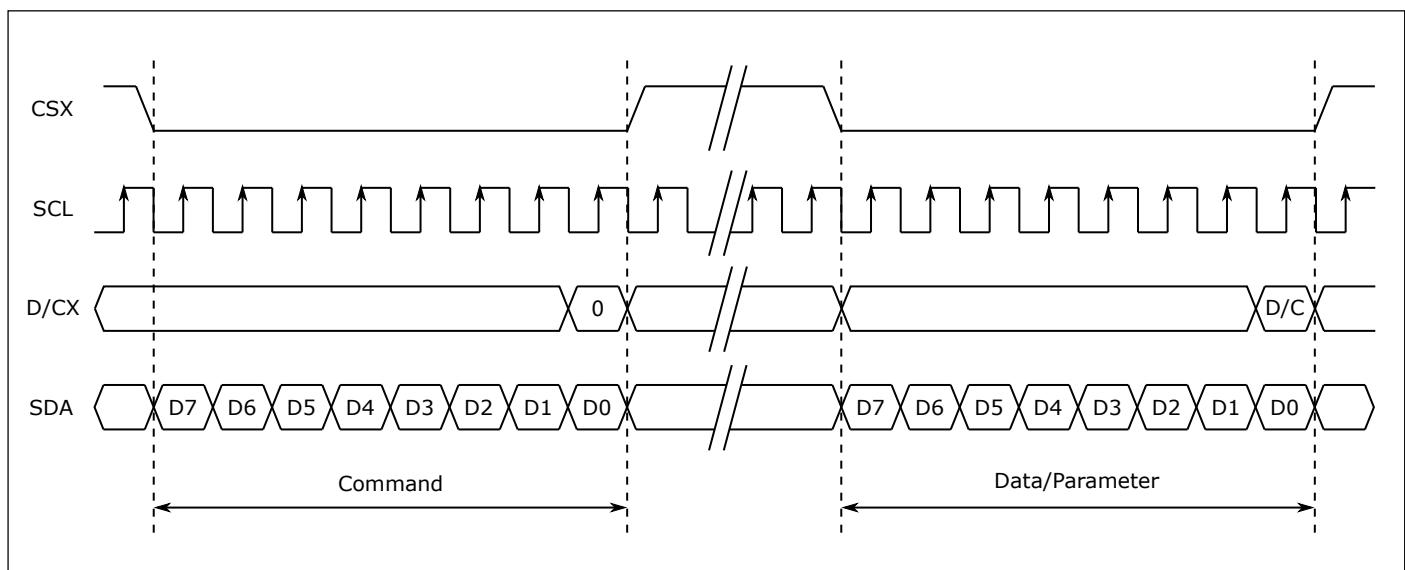


Fig. 22.12: Write timing

- CSX: Chip select signal. When the signal is low, the display module is selected, and when it is high, the display module will ignore all other interface signals

- D/CX: Data/Command selection signal. If the signal is low, the information transmitted by SDA is a command; if the signal is high, the information transmitted by SDA is display data or command parameters
- SCL: Serial clock input signal. Used to provide a clock signal for data transmission
- SDA: Serial data input/output signal. Used to transfer commands, command parameters, or data

22.3.3.2 Read timing

The sequence of MCU reading data from the display module is shown in the following figure:

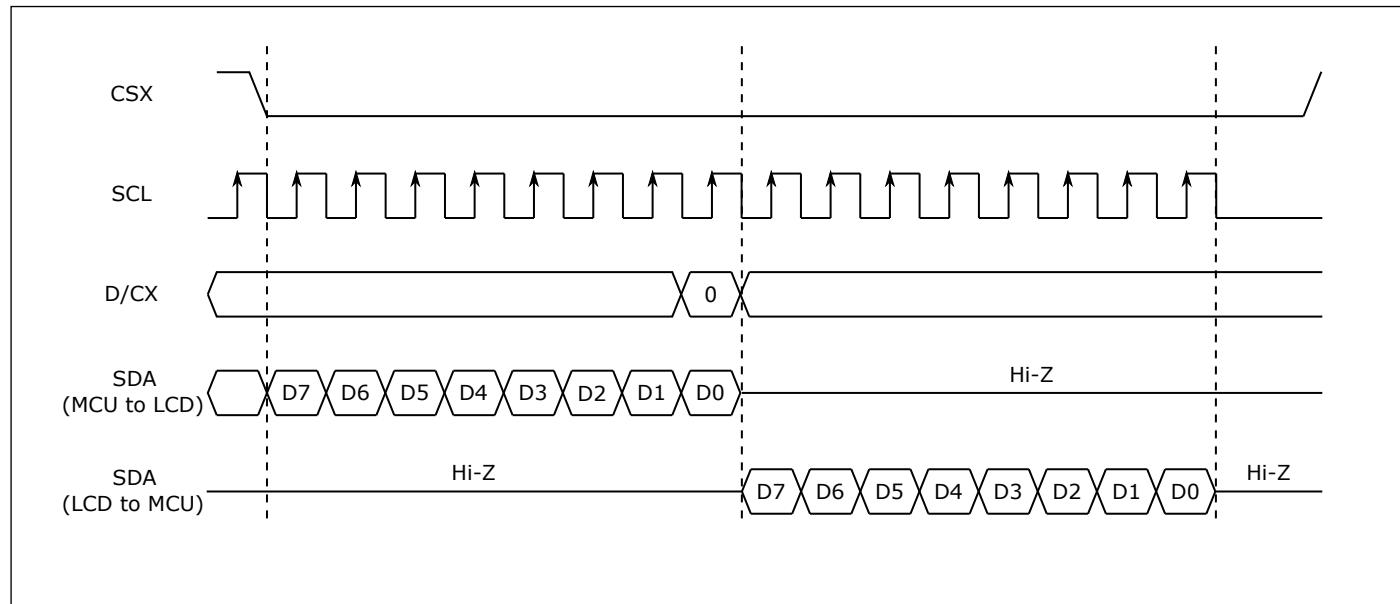


Fig. 22.13: Read timing

22.3.3.3 Output RGB565

RGB565 格式数据输出如下图所示：

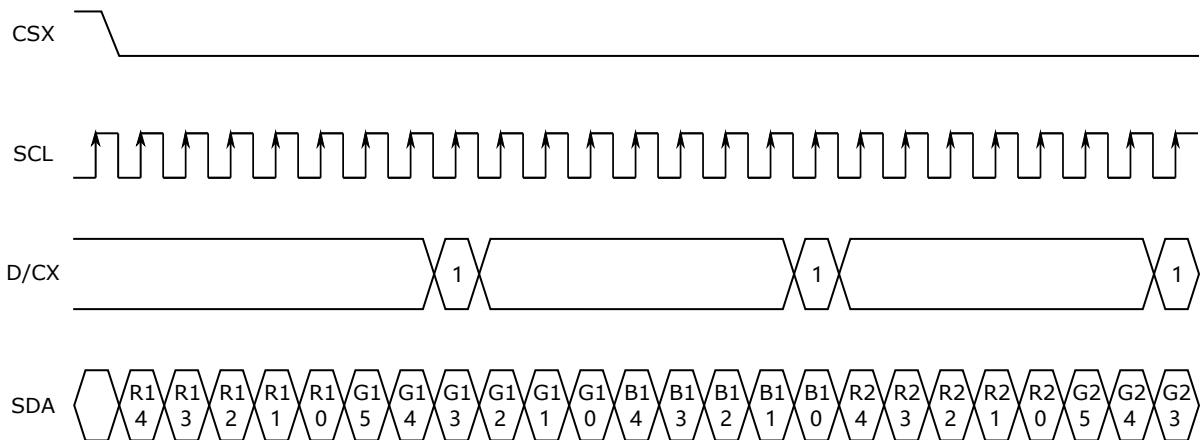


Fig. 22.14: RGB565 output

22.3.3.4 Output RGB666

The data output in RGB666 format is shown in the following figure:

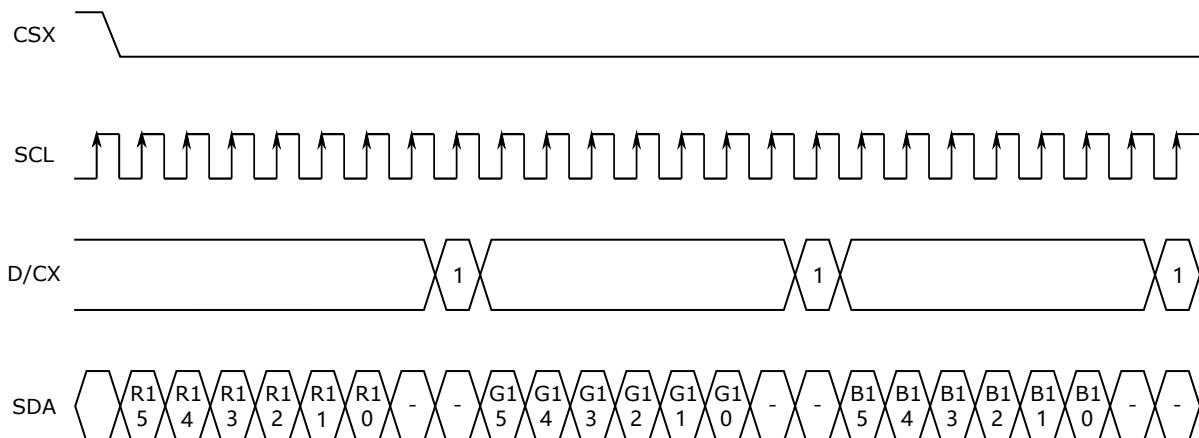


Fig. 22.15: RGB666 output

22.3.3.5 Output RGB888

The data output in RGB888 format is shown in the following figure:

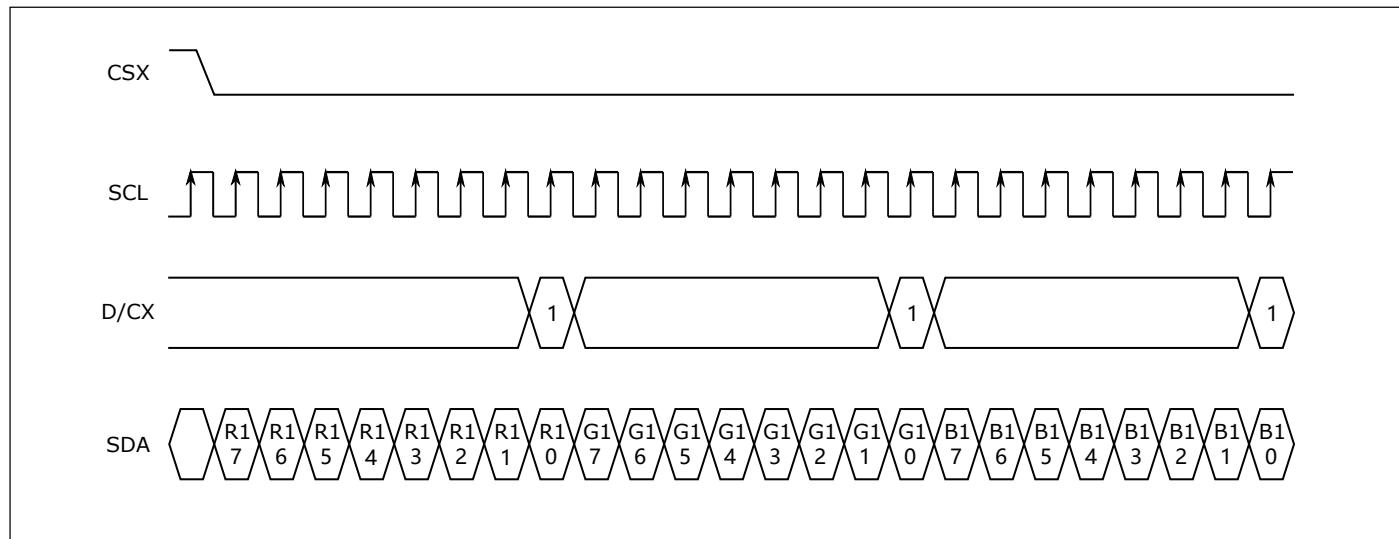


Fig. 22.16: RGB888 output

22.3.4 QSPI

22.3.4.1 Write timing

Taking 1-wire command, 1-wire 3-byte address, and 1-wire data pattern as examples, the timing of MCU writing data to the Display Module is shown as follows:

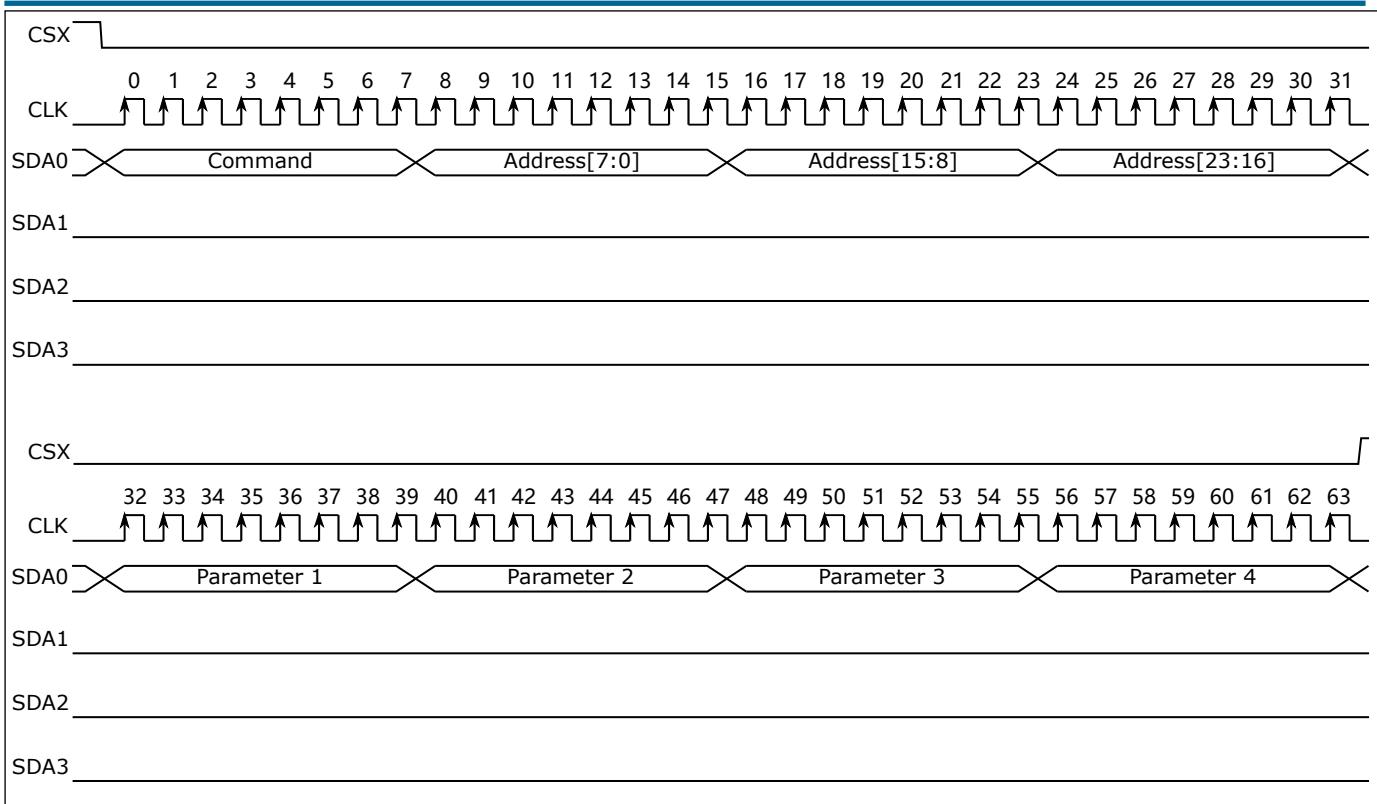


Fig. 22.17: Write timing

- CSX: chip select signal. When the signal is Low, the display module is selected; when it is High, the display module ignores all other interface signals.
- CLK: clock input signal, which provides a clock signal for data transfer.
- SDA0: The data line 0 is used to transmit commands, addresses or data.
- SDA1: The data line 1 is used to transmit commands, addresses or data.
- SDA2: The data line 2 is used to transmit commands, addresses or data.
- SDA3: The data line 3 is used to transmit commands, addresses or data.

22.3.4.2 Read timing

Taking 1-wire command, 1-wire 3-byte address, and 1-wire data pattern as examples, the timing of MCU reading data from the Display Module is shown as follows:

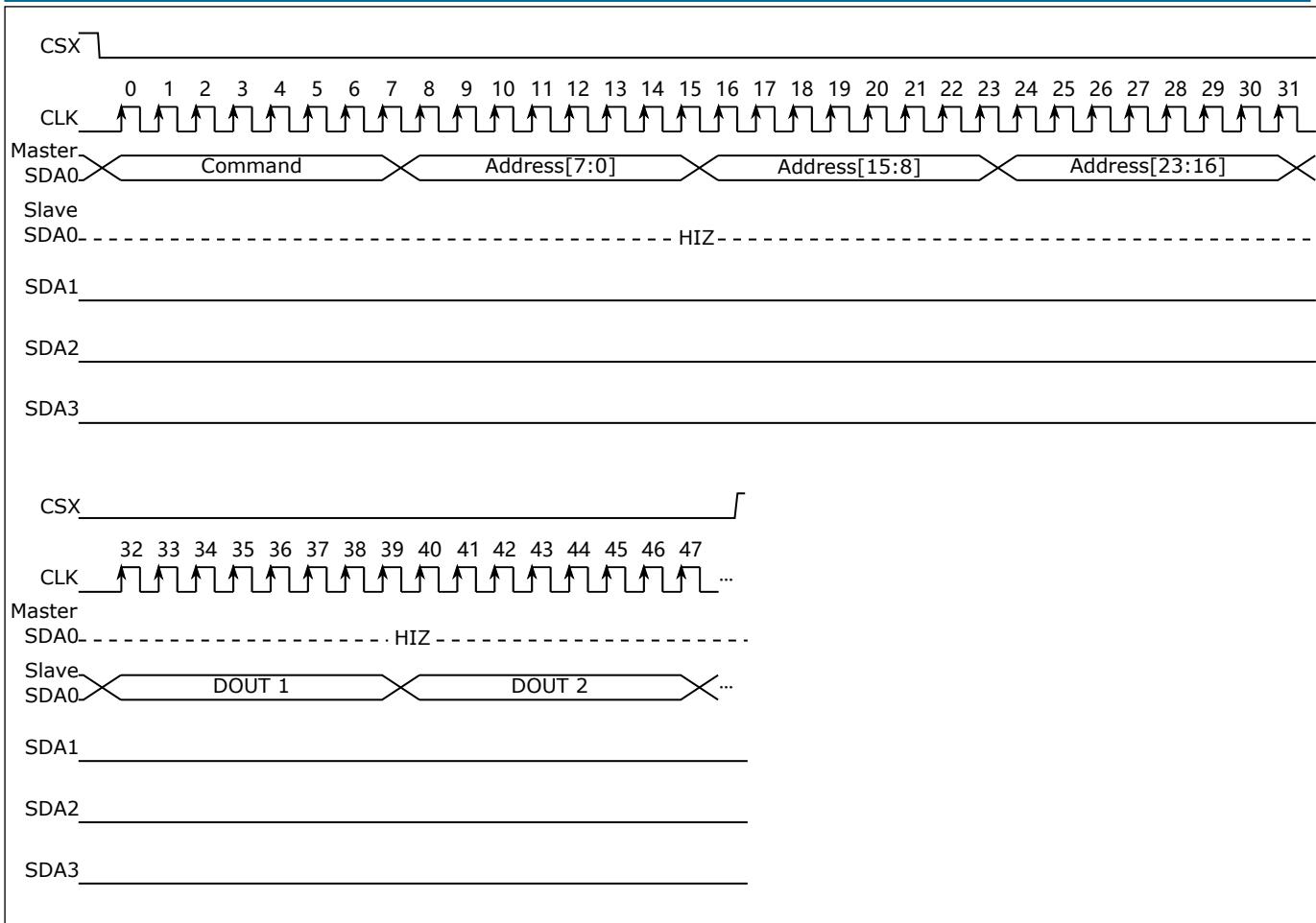


Fig. 22.18: Read timing

22.3.4.3 1-Wire Command, 1-Wire Address and 1-Wire Data Pattern

Taking the command 0x02 and 3-byte address 0x002C00/0x003C00 as example, the output of RGB565 data under 1-wire command, 1-wire address and 1-wire data pattern is shown as follows:

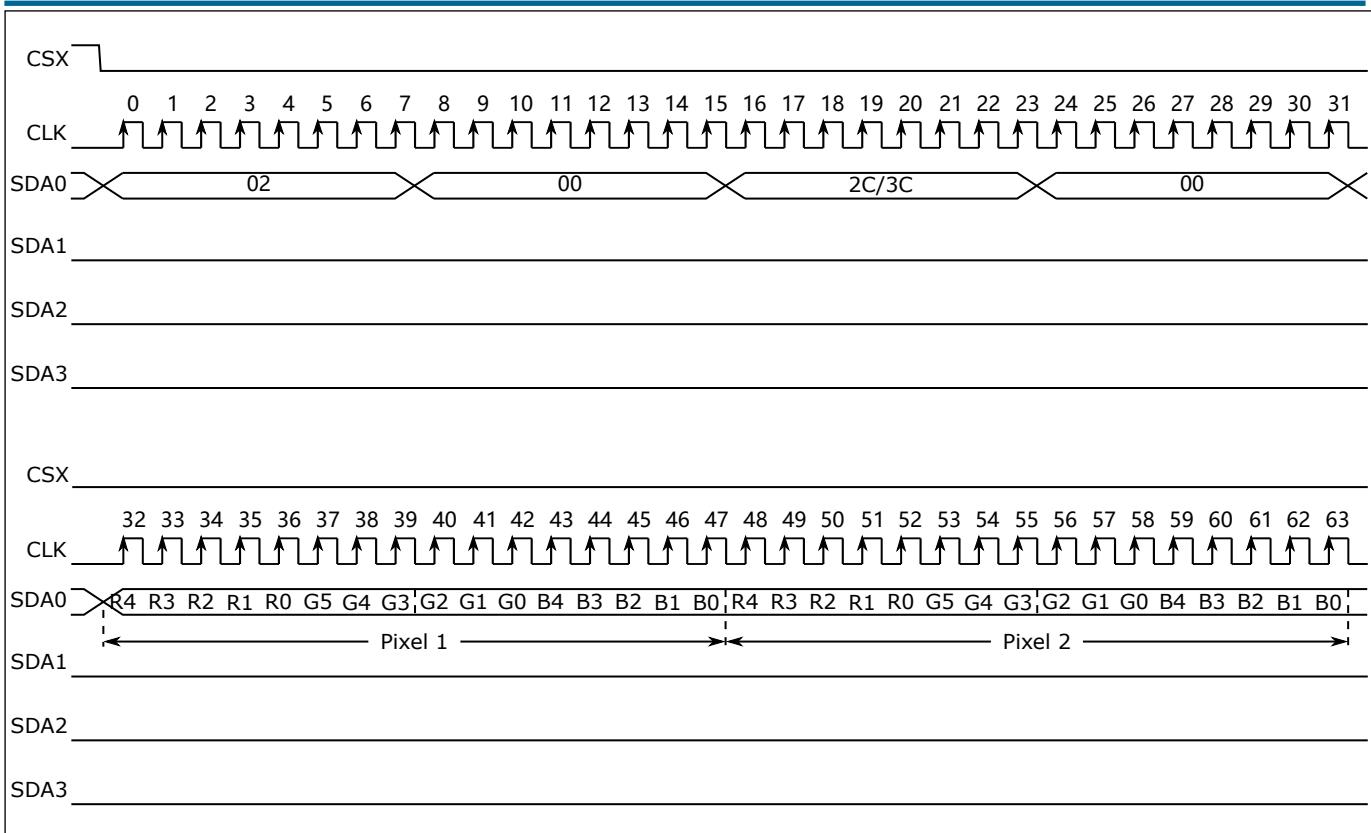


Fig. 22.19: RGB565 output

The data output in RGB666 format is shown as follows:

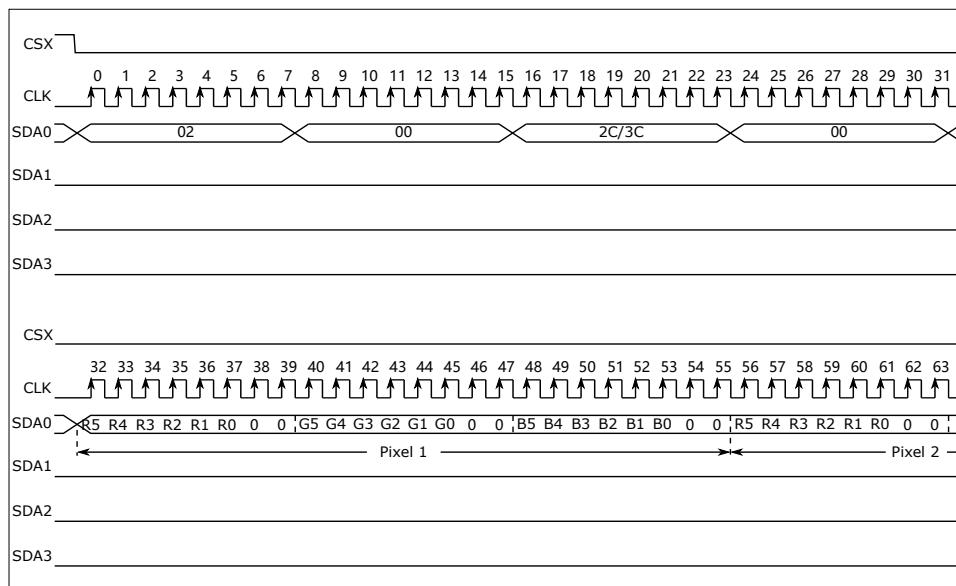


Fig. 22.20: RGB666 output

The data output in RGB888 format is shown as follows:

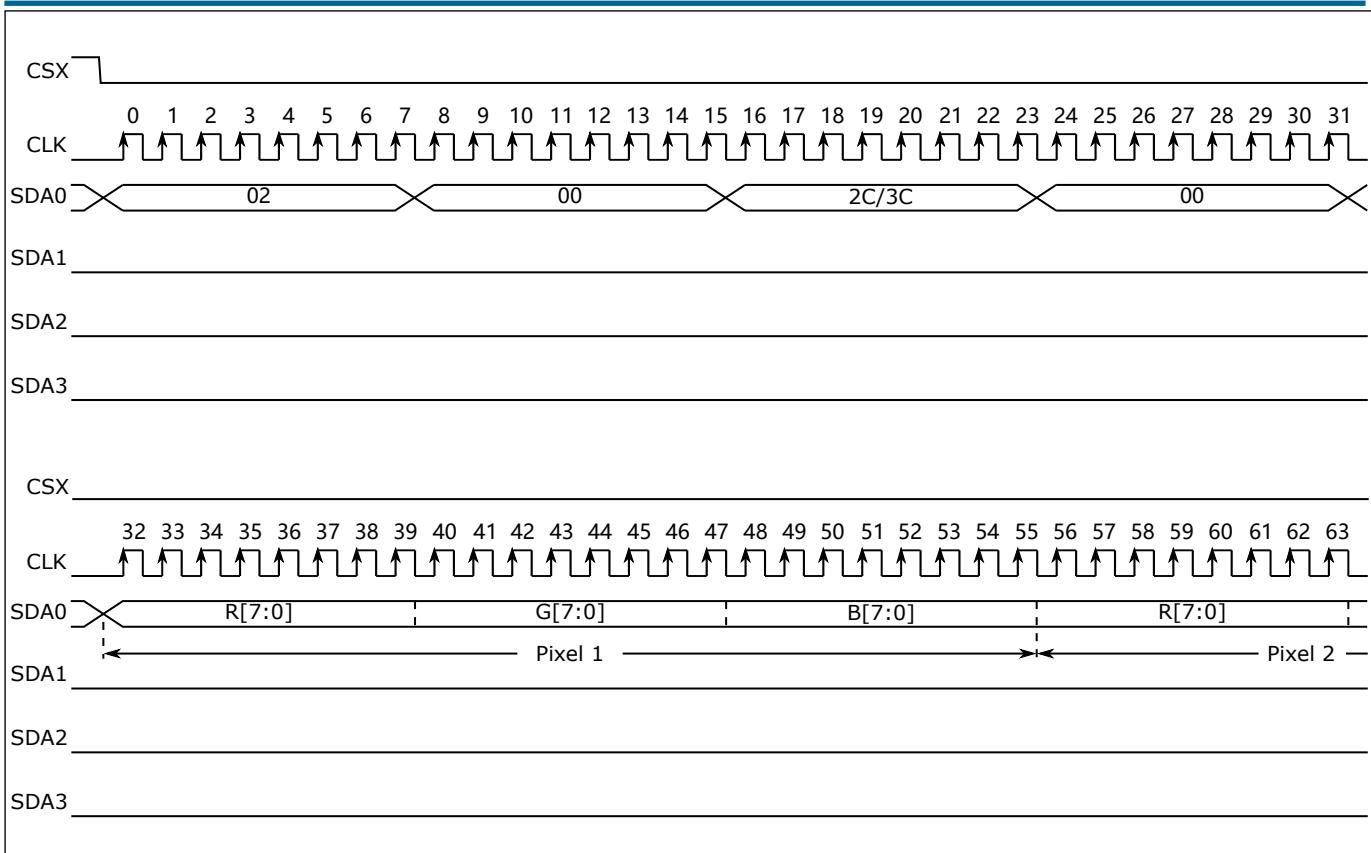


Fig. 22.21: RGB888 output

22.3.4.4 1-Wire Command, 1-Wire Address and 4-Wire Data Pattern

Taking the command 0x32 and 3-byte address 0x002C00/0x003C00 as example, the output of RGB565 data under 1-wire command, 1-wire address and 4-wire data pattern is shown as follows:

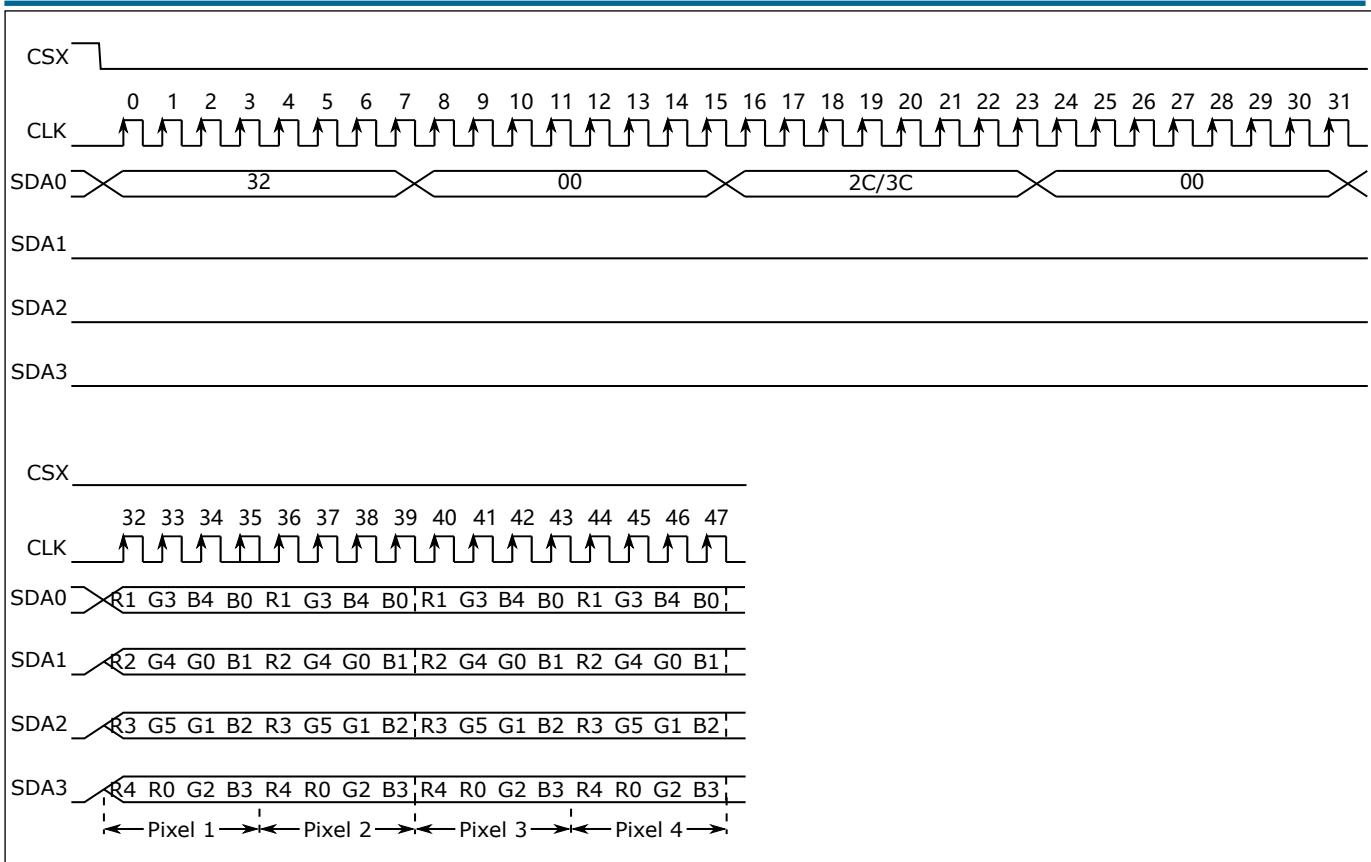


Fig. 22.22: RGB565 output

The data output in RGB666 format is shown as follows:

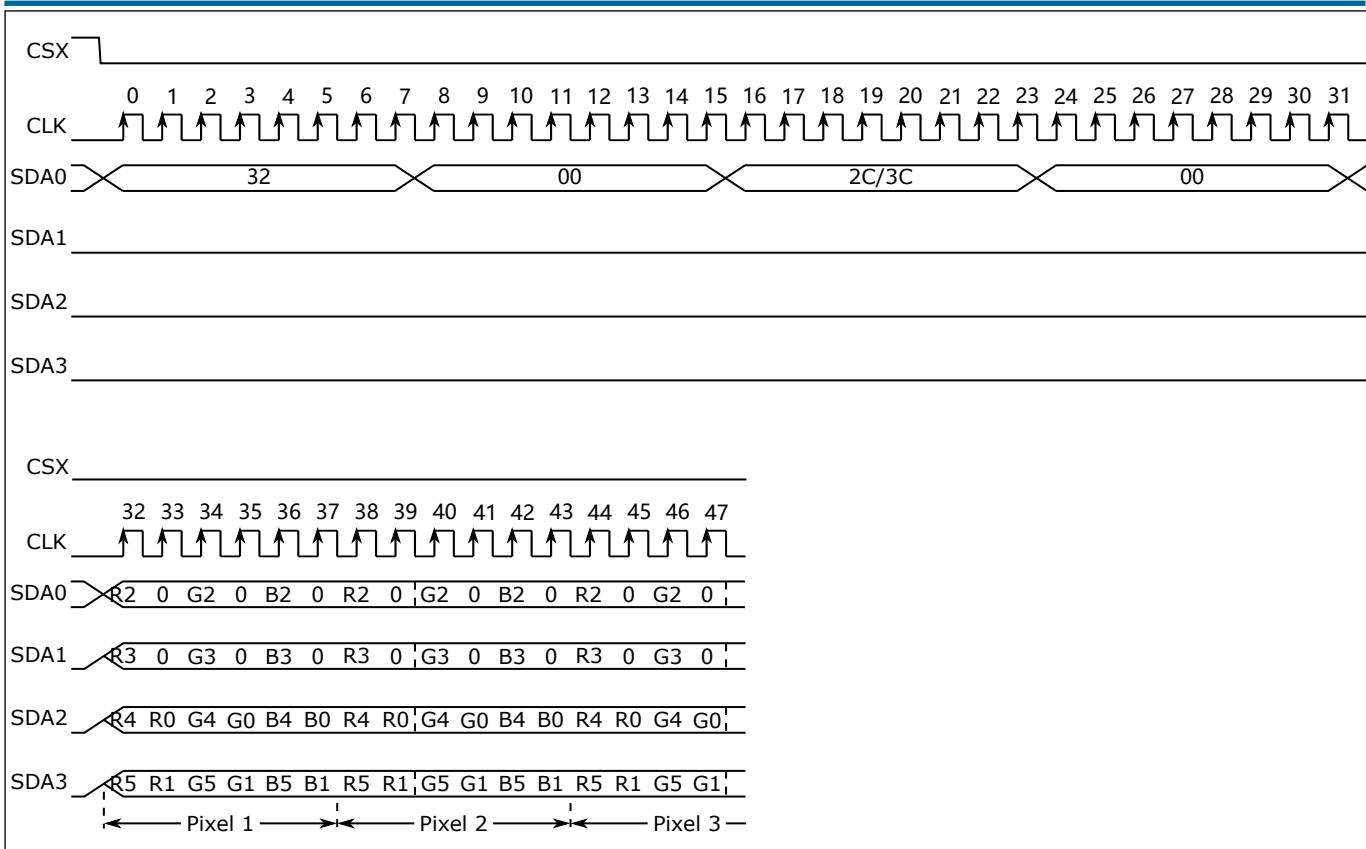


Fig. 22.23: RGB666 output

The data output in RGB888 format is shown as follows:

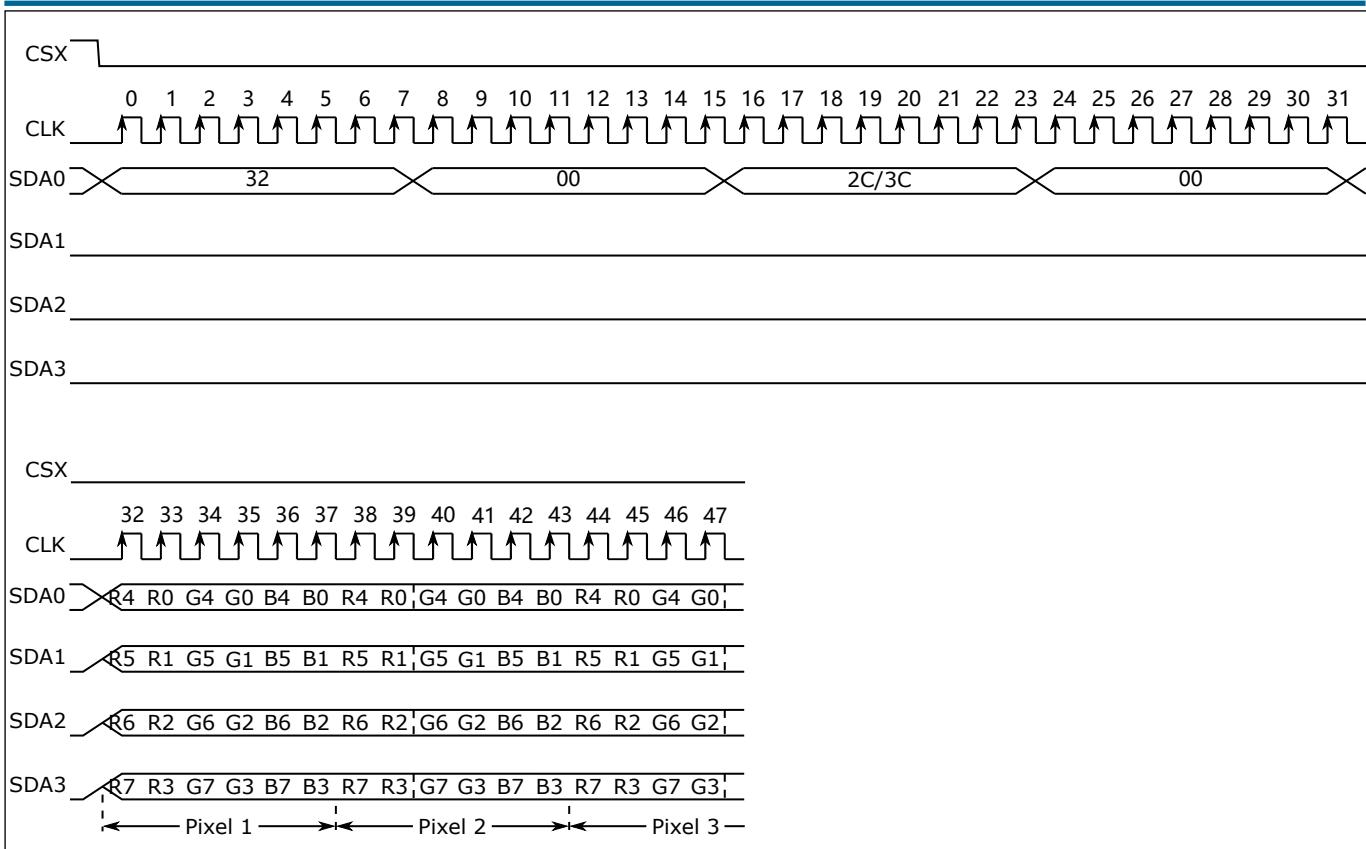


Fig. 22.24: RGB888 output

22.3.4.5 1-Wire Command, 4-Wire Address and 4-Wire Data Pattern

Taking the command 0x12 and 3-byte address 0x002C00/0x003C00 as example, the output of RGB565 data under 1-wire command, 4-wire address and 4-wire data pattern is shown as follows:

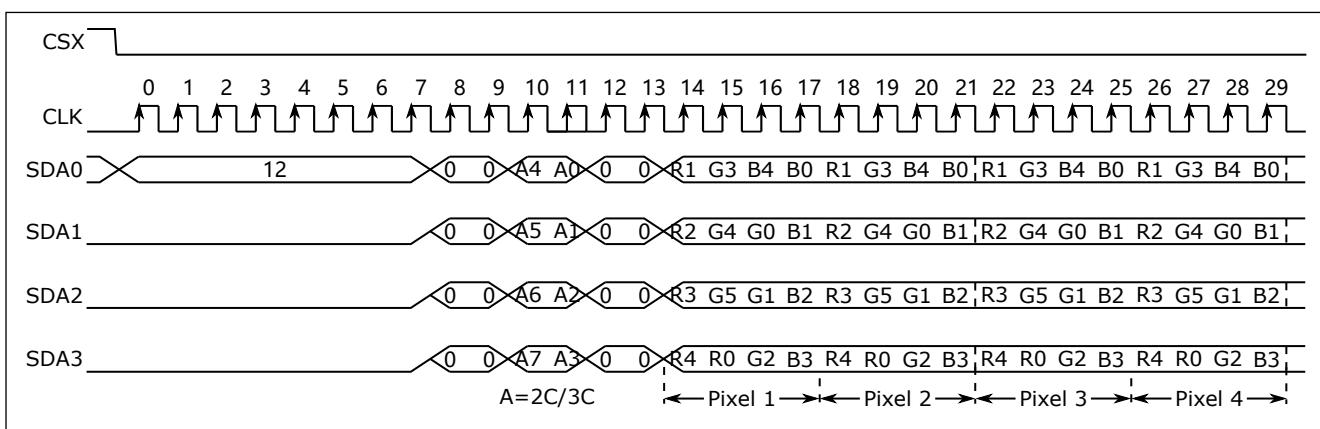


Fig. 22.25: RGB565 output

The data output in RGB666 format is shown as follows:

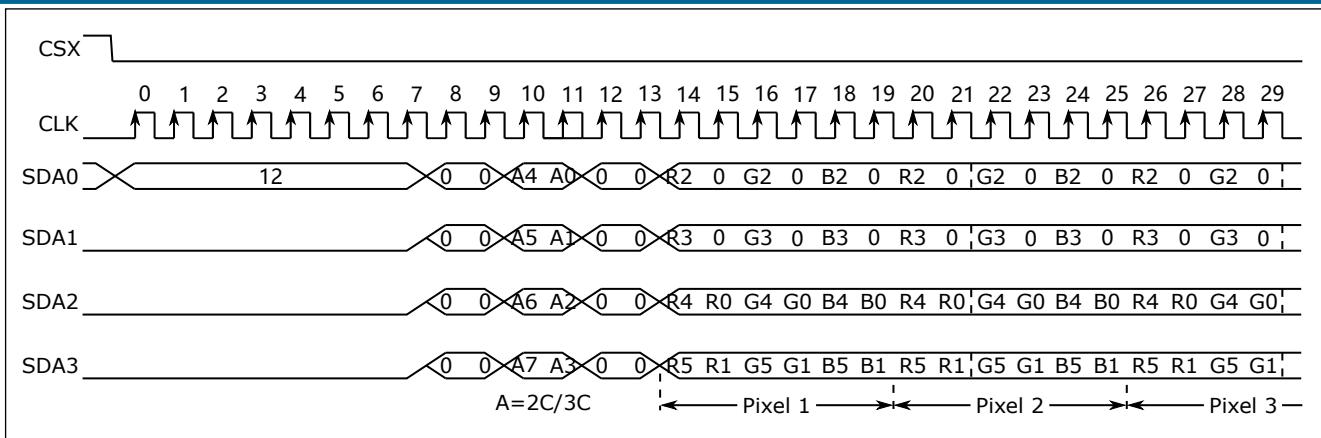


Fig. 22.26: RGB666 output

The data output in RGB888 format is shown as follows:

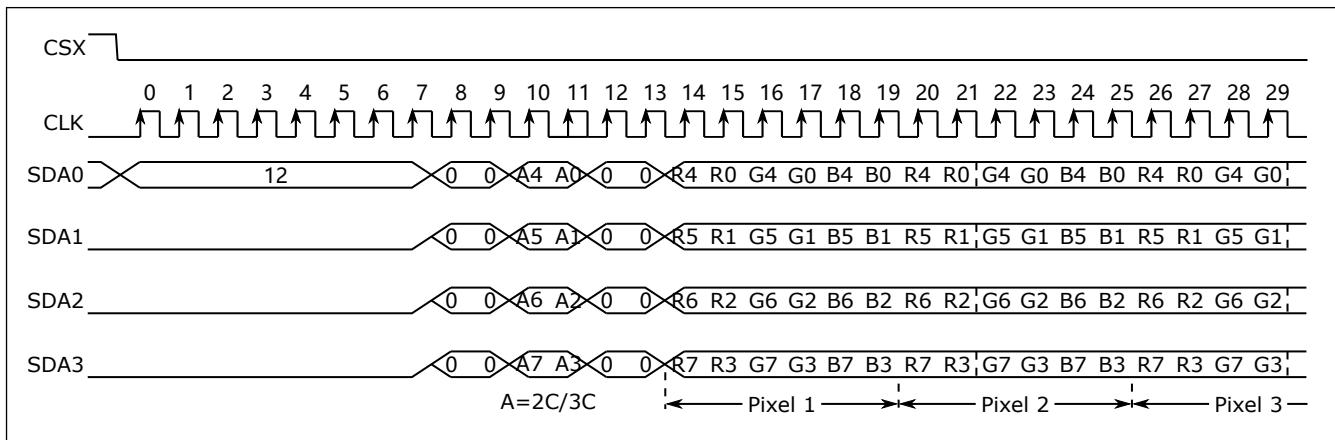


Fig. 22.27: RGB888 output

22.3.5 Input pixel format

The DBI module supports eight different input pixel RGB formats and six YUV444 formats. The RGB arrangements in the memory are shown as follows:

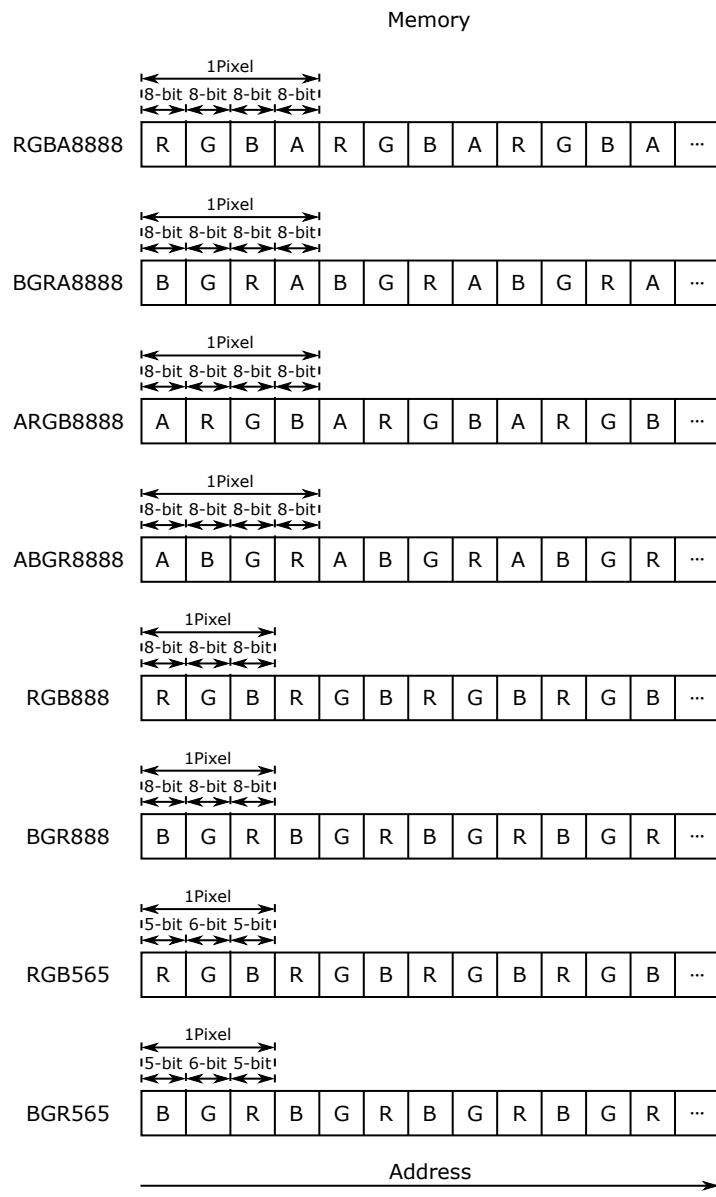


Fig. 22.28: Input Pixel RGB Format

For the YUV444 arrangement in the memory, just replace R with Y, G with U, and B with V.

22.3.6 Configuration of CS Signal Pull-up Release Condition

In Type B and Type C modes, the CS signal pull-up release condition can be configured by the CONT_EN bit in the register CONFIG. If this function cannot be enabled, the CS signal is released once after each pixel is transmitted. That is, the CS signal is pulled up every two pixels. If it is enabled, the CS signal will not be released during a single transmission. The CS signal will be pulled up and released only after this transmission is completed.

In QSPI mode, the CS signal pull-up release condition can be configured by the CS_STRETCH bit in the register CONFIG. If this function is disabled, during a single transmission, when FIFO is empty (that is, all the data in FIFO is sent out and no new data is filled in), the CS signal will be pulled up and released. If it is enabled, during a single transmission, when FIFO is empty (that is, all the data in FIFO is sent out and no new data is filled in), the CS signal will remain at a low level and wait for new data to be filled in.

22.3.7 Interrupt

DBI has the following interrupts:

- end of transfer interrupt
- TX FIFO request interrupt
- FIFO overrun error interrupt

By setting a pixel count value, the transmission end interrupt will be generated when the transmitted pixel data reaches the count value, and both Type B and Type C will generate this interrupt;

When TFICNT in DBI_FIFO_CONFIG_1 is greater than TFITH, a TX FIFO request interrupt is generated, and the interrupt flag will be automatically cleared when the condition is not met;

The FIFO overflow error interrupt will be generated when Overflow or Underflow occurs in the TX.

22.3.8 DMA

DBI TX supports DMA transfer mode, which requires setting the TX FIFO threshold through bit <TFITH> of register DBI_FIFO_CONFIG_1. When this mode is enabled, if TFICNT is greater than TFITH, a DMA TX request will be triggered. After the DMA is configured, when the DMA receives the request, it will transfer the data from the memory to the TX FIFO according to the settings.

23.1 Overview

The SD Host Controller (SDH) is used to control the communication with SDIO or SD card, and all accesses are set through standard control registers.

23.2 Features

- Comply with the SD Host Controller Specification defined by the SD Association
- Suitable for SDIO/SD memory card
- Supports the standard register settings for the SDH
- Supports the DMA operation for high-speed data transfer
- Supports interrupt
- Supports data block transmission

23.3 Functional Description

23.3.1 SDH Overall Structure

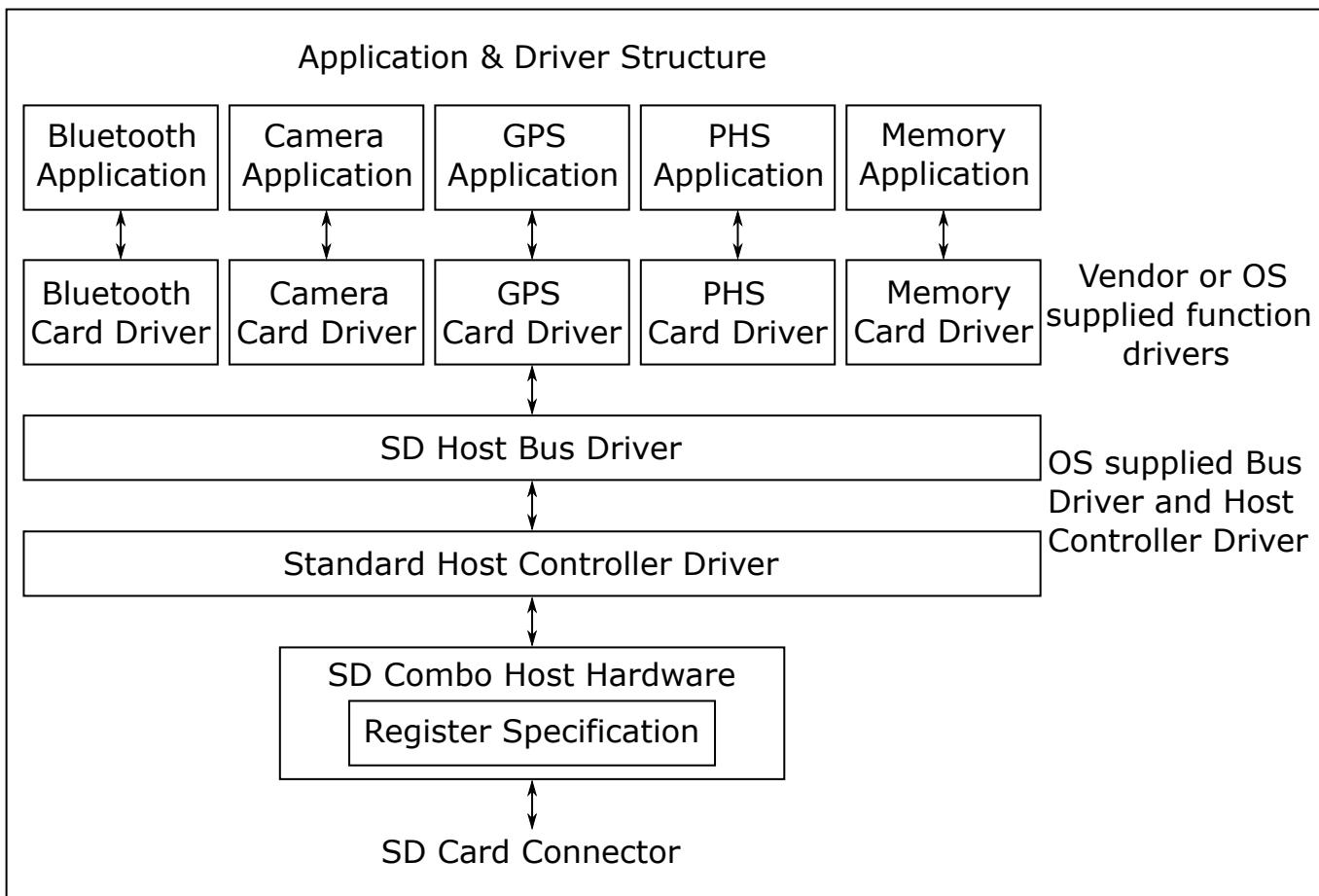


Fig. 23.1: SDH Hardware and Driver Structure

23.3.2 Register Mapping

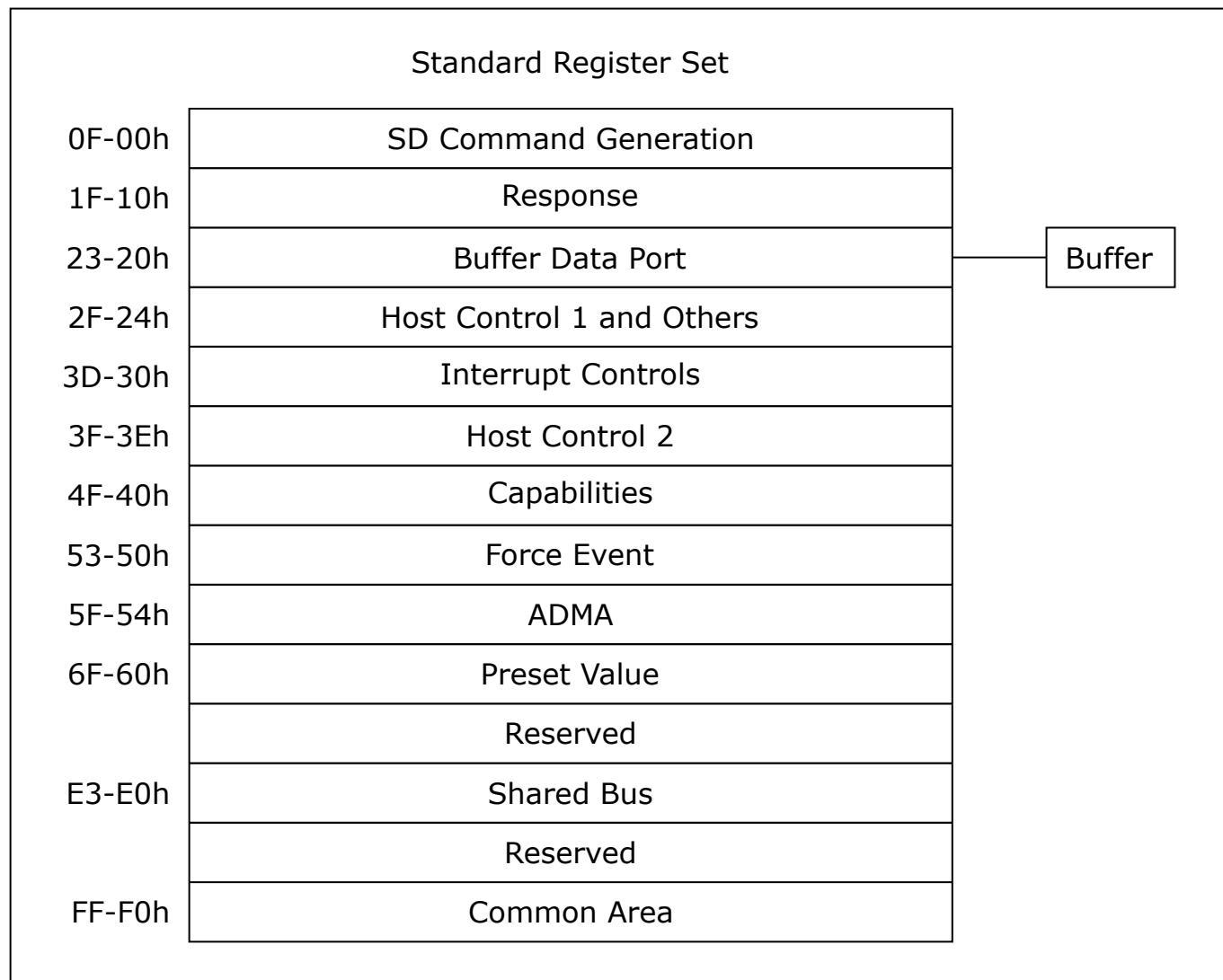


Fig. 23.2: Standard Register Mapping Classification

- SD Command Generation: parameter used to generate SD command
- Response: the response value from the card
- Buffer Data Port: the data access port of the internal buffer
- Host Control 1 and Others: current status, SD bus control, host reset, etc.
- Interrupt Controls: Interrupt Status and Interrupt Enable
- Host Control 2: vendor-specific host controller support information
- Capabilities: expansion of host control register
- Force Event: a test register that generates events through software

- ADMA: advanced DMA register
- Preset Value: preset values for clock frequency selection and driving strength selection
- Shared Bus: device control of shared bus system
- Common Area: public information area

23.3.3 Support for Multiple Card Slots

A standard register set is defined for each card slot. If the host controller has two card slots, two register sets are required. Each card slot is independently controlled. This makes it possible to support the combination of bus interface voltage, bus timing and SD clock frequency.

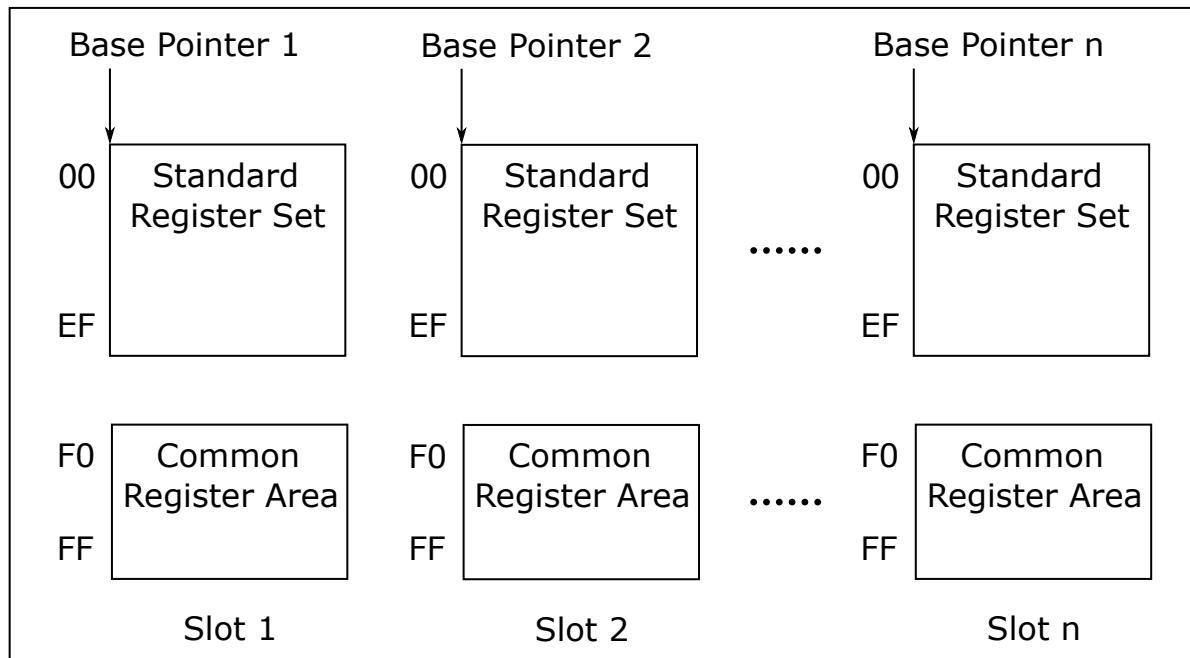


Fig. 23.3: Register Mapping of Multi-Card Slot Controller

The above figure shows the register mapping of the multi-card slot host controller. The host driver shall use PCI configuration registers or the vendor-specific methods to determine the number of card slots and the base pointer to the standard register set of each card slot. The offset from 0F0h to OFFh is reserved for the common register area, which defines the information of card slot control and public state. The common register area can be accessed from the register set of any card slot. This allows the software to control each card slot independently, because it can access the Card Slot Interrupt Status register and the Host Controller Version register from each register set.

23.3.4 Supporting DMA

The host controller provides a “programmed I/O” method for the host driver to transfer data using the Buffer Data Port register. Optionally, host controller implementers may support data transfer using DMA. Prior to using DMA, the host driver shall confirm that both the host controller and the system bus support it (PCI bus can support DMA). DMA shall support both single block and multiple-block transfers. The host controller registers shall remain accessible for issuing non-DAT line commands during a DMA transfer execution. The result of a DMA transfer shall be the same regardless of the system bus data transfer method. The host driver can stop and restart a DMA operation by the control bits in the block gap control register. By setting Stop At Block Gap Request, a DMA operation can be stopped at block gap. By setting Continue Request, DMA operation can be restarted. If an error occurs, DMA operation shall be stopped. To abort DMA transfer, the host driver shall reset the host controller through the “Software Reset” of DAT line in the software reset register, and issue CMD12 when executing read/write commands on multiple blocks.

23.3.5 SD Command Generation

	SDMA Command	ADMA Command	CPU Data Transfer	Non-DAT Transfer
SDMA System Address /Argument 2	Yes /No	No /Auto CMD23	No /Auto CMD23	No /No
Block Size	Yes	Yes	Yes	No (Protected)
Block Count	Yes	Yes	Yes	No (Protected)
Argument 1	Yes	Yes	Yes	Yes
Transfer Mode	Yes	Yes	Yes	No (Protected)
Command	Yes	Yes	Yes	Yes

Fig. 23.4: Register for Generating SD Commands

The table above shows the register settings (offset from 000h to 00Fh in a register set) required for three types of transactions: DMA-generated transfer (SDMA or ADMA), CPU data transfer (using “programmed I/O”) and non-DAT transfer. When starting a transaction, the host driver shall program these registers from 000h to 00Fh. The offset of the starting register can be calculated based on the transaction type. The last written offset shall always be 00Fh, because the high byte written to the command register will trigger the SD command. During a data transaction, the host driver shall not read the SDMA system address, block size and block count registers, unless transfer is stopped or suspended because the values are changing and unstable. To prevent data transfer from damaging registers when commands are issued, the block size, block count and transfer mode registers shall be write-protected by the host controller, and the command disable (DAT) shall be set to 1 in the current status register (this signal cannot protect the SDMA system address). When command disable (CMD) is set to 1, the host driver must not write parameter 1

and command register.

23.3.6 Suspend and Resume Mechanism

Support for Suspend/Resume can be determined by checking Suspend/Resume Support in the Capabilities register. When the SD card accepts a suspend request, the host driver saves information in the first 14 bytes registers (that is, offsets 000h-00dh) before issuing other SD commands. On resuming, the host driver restores these registers and then issues the Resume command to continue suspended operation. The SDIO card sets the DF (Resume Data Flag) in the response to the Resume command. (Since the Suspend and Resume commands are CMD52 operations, the response data is actually the Function Select Register in the CCCR.) If DF is set to 0, it means that the SDIO card cannot continue data transfer while suspended. This bit can be used to control data transfers and interrupt cycles. If the Resume Data Flag is set to 0, no more data is transferred and an interrupt cycle is started if the transaction being resumed is in 4-bit mode. If DF is set to 1, data transfers continue. It is worth noting that to use Suspend/Resume function, the SDIO card must support Suspend and Resume commands and Read Wait control.

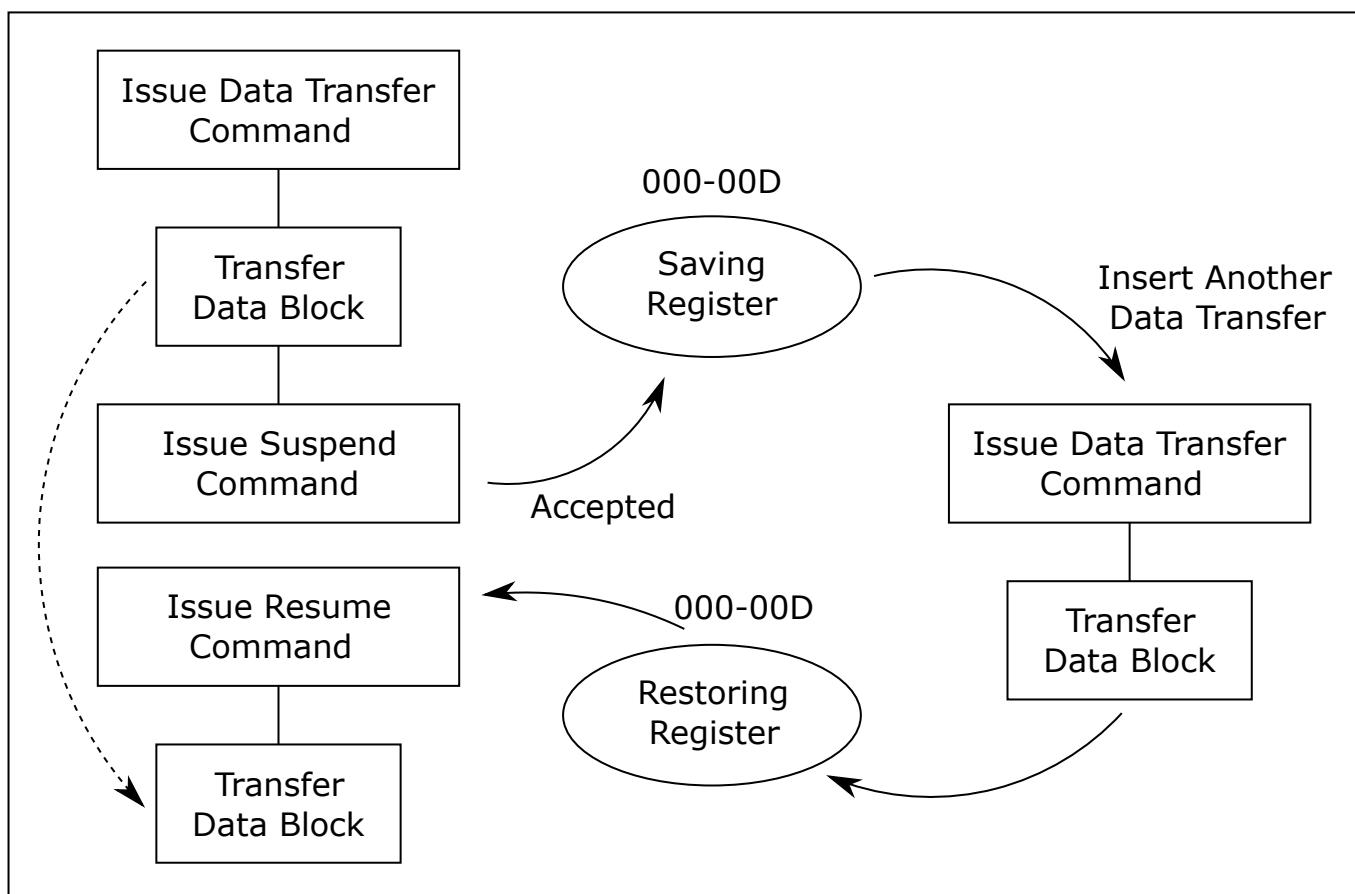


Fig. 23.5: Suspend and Resume Mechanism

23.3.7 Buffer Control

The host controller has a data buffer for data transfer. The host driver accesses internal buffer through the 32-bit Buffer Data Port register. Here are some rules for accessing the buffer.

23.3.7.1 Control of Buffer Pointer

Internally, the host controller maintains a pointer to control the data buffer. The pointer is not directly accessible by the host driver. Every time the Buffer Data Port register is accessed, the pointer is incremented depending on amount of data written to the buffer. In order to accommodate a variety of system buses, this pointer shall be implemented regardless of system bus width (8/16/32/64-bit system bus width can be supported).

23.3.7.2 Determination of Buffer Block Length

To transfer data blocks at a burst, the relationship between host controller and SD card buffer sizes is very important. The host driver shall use the same data block length for both host controller and SD card. If the controller and card buffer sizes are different, the host driver shall use the smaller one. The maximum host controller buffer size is defined by the Max Block Length field in the Capabilities register.

23.3.7.3 Dividing Large Data Transfer

The SDIO command CMD53 defines the maximum data size for transfer according to the following formula: Maximum data size = block size x block count. For example, the block size is specified by the buffer size, and the maximum value of block count is 512 (9-bit count), as specified in the command parameter CMD53. In the worst case, if the card only has a 1-byte buffer, CMD53 can be used to transfer 512 bytes of data at most (block size = 1, block count = 512). If the card does not support the multi-block mode, only one byte can be transferred. If an application or card driver wants to transfer data with a larger size, the host driver shall divide the large-size data into multiple CMD53 blocks.

23.3.8 Relationship Between Interrupt Control Registers

The host controller implements a number of interrupt sources. Interrupt sources can be enabled as interrupts or as system wakeup signals. If the interrupt source's corresponding bit in the Normal Interrupt Status Enable or Error Interrupt Status Enable register is 1 and the interrupt becomes active, its active state is latched and made available to the host driver in the Normal Interrupt Status register or the Error Interrupt Status register. Interrupt Status shall be cleared when Interrupt Status Enable is cleared. An interrupt source with its bit set in an Interrupt Status register shall assert a system interrupt signal if its corresponding bit is also set in the Normal Interrupt Signal Enable register or the Error Interrupt Signal Enable register. Once signaled, most interrupts are cleared by writing a 1 to the associated bit in the Interrupt Status register. Card interrupts, however, shall be cleared by the card driver. If a card interrupt is generated, the host driver may clear Card Interrupt Status Enable to disable card interrupts while the card driver is

processing them. After all interrupt sources are cleared, the host driver sets it again to enable another card interrupt. Disabling the Card Interrupt Status Enable avoids generating multiple interrupts during interrupt service processing. The Wakeup Control register enables Card Interrupt, Card Insertion, or Card Removal status changes to be configured to generate a system wakeup signal. These interrupts are enabled or masked independently of the Normal Interrupt Signal Enable register. The kind of wakeup event can be read from the Normal Interrupt Status register. The interrupt signal and wakeup signal are logical ORed and shall be read from the Slot Interrupt Status register.

23.3.9 Hardware Block Diagram and Timing Part

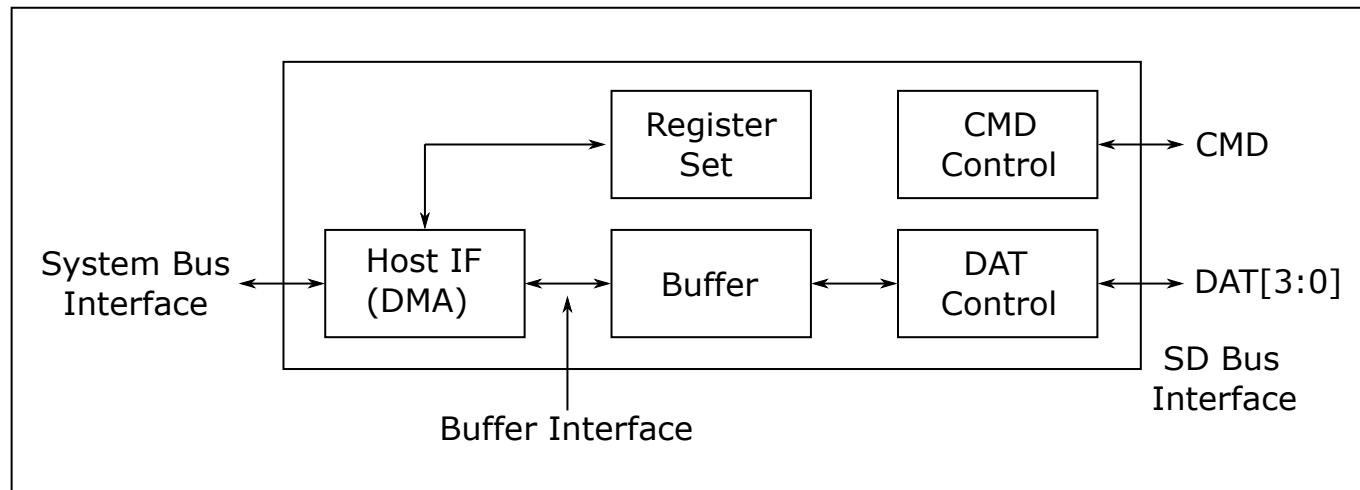


Fig. 23.6: Block Diagram of Host Controller

The host controller has two bus interfaces, the system bus interface and the SD bus interface. The host controller assumes that these interfaces are asynchronous, that is, working at different clock frequencies. The host driver is on system bus time, because it is software executed by the host controller CPU on its system clock. The SD card is on SD bus time. That is, its operation is synchronized by SDCLK. The host controller shall synchronize signals to communicate between these interfaces. Data blocks shall be synchronized at the buffer module. All status registers shall be synchronized by the system clock and maintain synchronization during output to the system interface. Control registers, which trigger SD bus transactions, shall be synchronized by SDCLK. Therefore, there will be a timing delay when propagating signals between the two interfaces. This means that the host driver cannot do real time control of the SD bus and needs to rely on the host controller to control the SD bus according to register settings. The buffer interface enables internal read and write buffers. The transfer complete interrupt status indicates completion of the read/write transfer for both DMA and non-DMA transfers. However, this timing is different between reads and writes. Read transfers shall be completed after all valid data have been transferred to the host system and are ready for the host driver to access. Write transfers shall be completed after all valid data have been transferred to the SD card and the busy state is over.

23.3.10 Auto CMD12

Multiple block transfers for SD memory require CMD12 to stop the transactions. The host controller automatically issues CMD12 when the last block transfer is completed. This feature of the host controller is called Auto CMD12. The host driver shall set Auto CMD12 Enable in the Transfer Mode register when issuing a multiple block transfer command. Auto CMD12 timing synchronization with the last data block shall be done by hardware in the host controller. Commands that do not use the DAT line can be issued during multiple block transfers. These commands are referred to using the notation CMD_wo_DAT. To prevent DAT line commands and CMD_wo_DAT commands from conflicting, the host controller shall arbitrate the timing by which each command is issued on the SD Bus. Therefore, a command might not immediately be issued after the host driver writes to the Command register. The command may be issued before or after Auto CMD12, depending on the timing. To be able to distinguish the responses of DAT line and CMD_wo_DAT commands, the Auto CMD12 response can be determined from the upper four bytes of the Response register (at offset 01Ch in the standard register set). If errors related to Auto CMD12 are detected, the host controller shall issue an Auto CMD error interrupt. The host driver can check the Auto CMD12 error status (Command Index/End bit/CRC/Timeout Error) by reading the Auto CMD Error Status register. If the Auto CMD12 was not executed, the host driver needs to recover from the CMD_wo_DAT error and issue CMD12 to stop the multiple block transfer. If the CMD_wo_DAT was not executed, the host driver can issue it again after recovering from the Auto CMD12 error. In UHS mode SDR104, the host driver shall use Auto CMD23 to stop multiple block read/write operation instead of using Auto CMD12. In other bus speed modes, if the card supports CMD23, the host driver shall use Auto CMD23 instead of using CMD12.

23.3.11 Controlling SDCLK

This table shows how SDCLK is controlled by the SD Bus Power in the Power Control register and the SD Clock Enable in the Clock Control register.

SD Bus Power	SD Clock Enable	State of SDCLK
Change 0 to 1	0	Drive Low
	1	Start Clock With Specified Period Of High
Change 1 to 0	0	Drive Low
	1	Drive Low Immediately
0	Don't Care	Drive Low
1	Change 0 to 1	Start Clock With Specified Period Of High
	Change 1 to 0	Maintains Period Of High And Then Stops Clock And Drive Low

Fig. 23.7: Controlling SDCLK by the SD Bus Power and SD Clock Enable

The clock period of SDCLK is specified by the SDCLK Frequency Select in the Clock Control register and the Base Clock Frequency For SD Clock in the Capabilities register. Since the SD card may use both clock edges, the duty

ratio of SD clock shall be average 50% (scattering within 45-55%) and the Period of High should be half of the clock period. The oscillation of SDCLK starts from driving specified Period of High. When SDCLK is stopped by the SD Clock Enable, the host controller shall stop SDCLK after driving Period of High to maintain the duty ratio of clock. When SDCLK is stopped by the SD Bus Power, the host controller shall stop SDCLK immediately (drive Low) and SD Clock Enable shall be cleared.

23.3.12 Advanced DMA

The SD Host Controller Standard Specification Version 2.00 defines the advanced DMA (ADMA) transfer algorithm. The DMA algorithm defined in the SD Host Controller Standard Specification Version 1.00 is called single DMA (SDMA). The disadvantage of SDMA is that the DMA interrupt generated at every page boundary disturbs CPU to reprogram the new system address. This SDMA algorithm develops a performance bottleneck by interrupting at each page boundary. ADMA adopts the scatter gather DMA algorithm so that higher data transfer speed is available. Before executing ADMA, the host driver can program a list of data transfers between system memory and SD card to the descriptor table. It enables ADMA to run without interrupting the host driver. In addition, ADMA supports both 32-bit system memory addressing and 64-bit system memory addressing. The 32-bit system memory addressing uses the lower 32-bit field of the 64-bit address register. ADMA is divided into ADMA1 and ADMA2. ADMA1 only supports the transfer of 4KB aligned data in the system memory. ADMA2 optimizes this restriction, so that the data of any size at any location can be transferred in the system memory. The formats of descriptor tables are different among them. In this document, the term “ADMA” refers to ADMA2.

23.3.12.1 Block Diagram of ADMA2

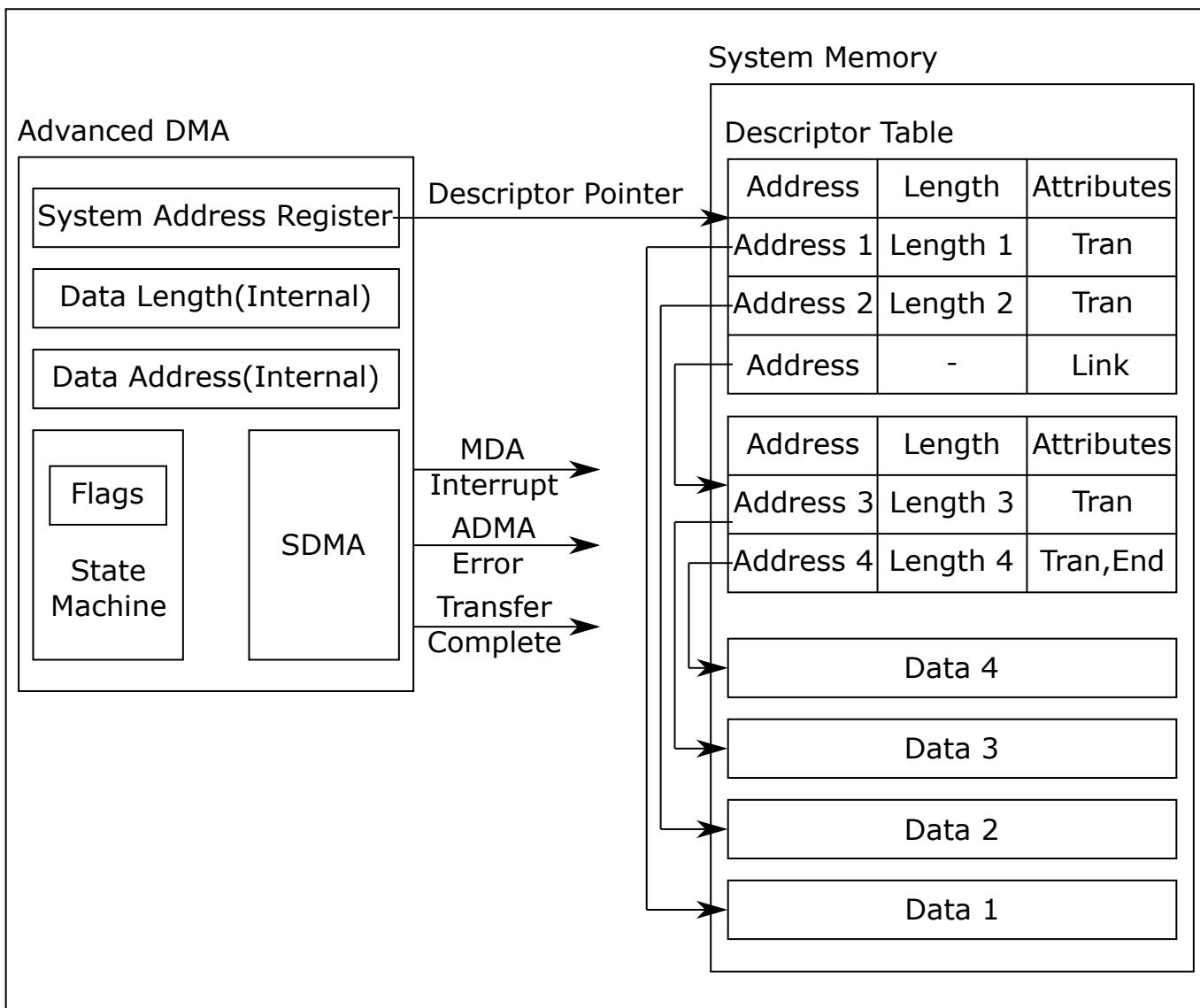


Fig. 23.8: Block Diagram of ADMA2

The descriptor table is created by the host driver in the system memory. The 32-bit address descriptor table is used for the 32-bit addressing system, and 64-bit address descriptor table is used for the 64-bit addressing system. Each descriptor line (one executable unit) consists of address, length and attribute fields. This attribute specifies the operation of the descriptor line. ADMA2 consists of SDMA, state machine and register circuit. ADMA2 uses the 64-bit advanced DMA system address register (offset 058h) as the descriptor pointer, instead of the 32-bit SDMA system address register (offset 0). Writing to the command register triggers the termination of ADMA2 transfer. ADMA2 takes a descriptor line and executes it. This process is repeated until the end of the descriptor (the attribute “end” equals to 1) is found.

23.3.12.2 Data Address and Data Length Requirements

There are three requirements for programming descriptors:

- The minimum unit of address is 4 bytes.
- The maximum data length of each descriptor line is less than 64 KB.
- Total length = length 1 + length 2 + length 3 + ⋯ + length n = multiple of block size.

If the total length of the descriptor is not a multiple of the block size, the ADMA2 transfer may not be terminated. In this case, the transfer shall be terminated by data timeout. The block count register allows the transfer of maximum 65,535 blocks. If the ADMA2 operation is less than or equal to the transfer of 65,535 blocks, the block count register can be used. In this case, the total length of the descriptor table shall be equal to the product of the block size and the block count. If the ADMA2 operation exceeds the transfer of 65,535 blocks, the block count register shall be disabled by setting Block Count Enable to 0 in the transfer mode register. In this case, the length of data transfer is not specified by the block count but by the descriptor table. Thus, the timing of detecting the last block on the SD bus may be different, and it affects the control of read transfer activity, write transfer activity and DAT line activity in the current status register. In the case of a read operation, multiple blocks may be read. If the read operation is for the last memory area, the host driver shall ignore the OUT_OF_RANGE error.

23.3.12.3 Descriptor Table

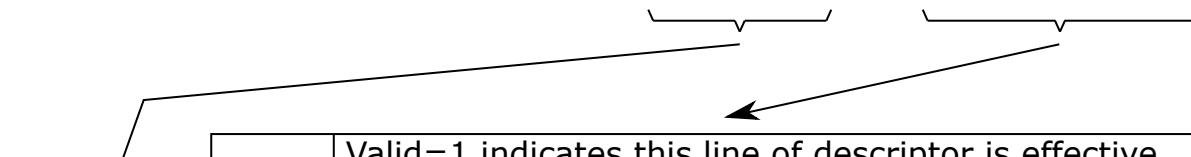
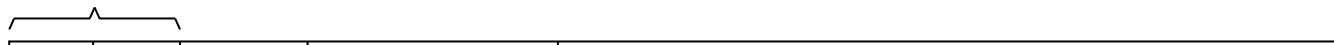
Address Field		Length		Reserved		Attribute					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address	16-bit Length	000000	Act2	Act1	0	Int	End	Valid			
											
Valid Valid=1 indicates this line of descriptor is effective. If Valid=0 generate ADMA Error Interrupt and stop ADMA to prevent runaway.											
End End=1 indicates end of descriptor. The Transfer Complete Interrupt is generated when the operation of the descriptor line is completed.											
Int INT=1 generates DMA Interrupt when the operation of the descriptor line is completed.											
											
Act2	Act1	Symbol	Comment	Operation							
0	0	Nop	No Operation	Do not execute current line and go to next line.							
0	1	Rsv	Reserved	Do not execute current line and go to next line.							
1	0	Tran	Transfer Data	Transfer data of one descriptor line.							
1	1	Link	Link Descriptor	Link to another descriptor.							

Fig. 23.9: 32-Bit Address Descriptor Table

The above figure shows the definition of the 32-bit address descriptor table. One descriptor line consumes 64 bits (8 bytes) of memory space. The attribute is used to control descriptors. Three action symbols are specified here. The “Nop” operation skips the current descriptor line and goes to the next line. The “Tran” operation transfers the data specified by the address and length fields. The “Link” operation is used to link two separate descriptors. The link address field points to the next descriptor table. The combination of Act2=0 and Act1=1 is reserved and defined as the same operation as “Nop”. Future versions of the controller may use this field and redefine new operations. The 32-bit address is stored in the lower 32 bits of the 64-bit address register. For a 32-bit address descriptor table, the address field shall be set on the 32-bit boundary (the lower 2 bits are always set to 0).

23.3.12.4 ADMA2 State

This figure illustrates the four states of ADMA2: fetch descriptor state, change address state, transfer data state and stop ADMA state:

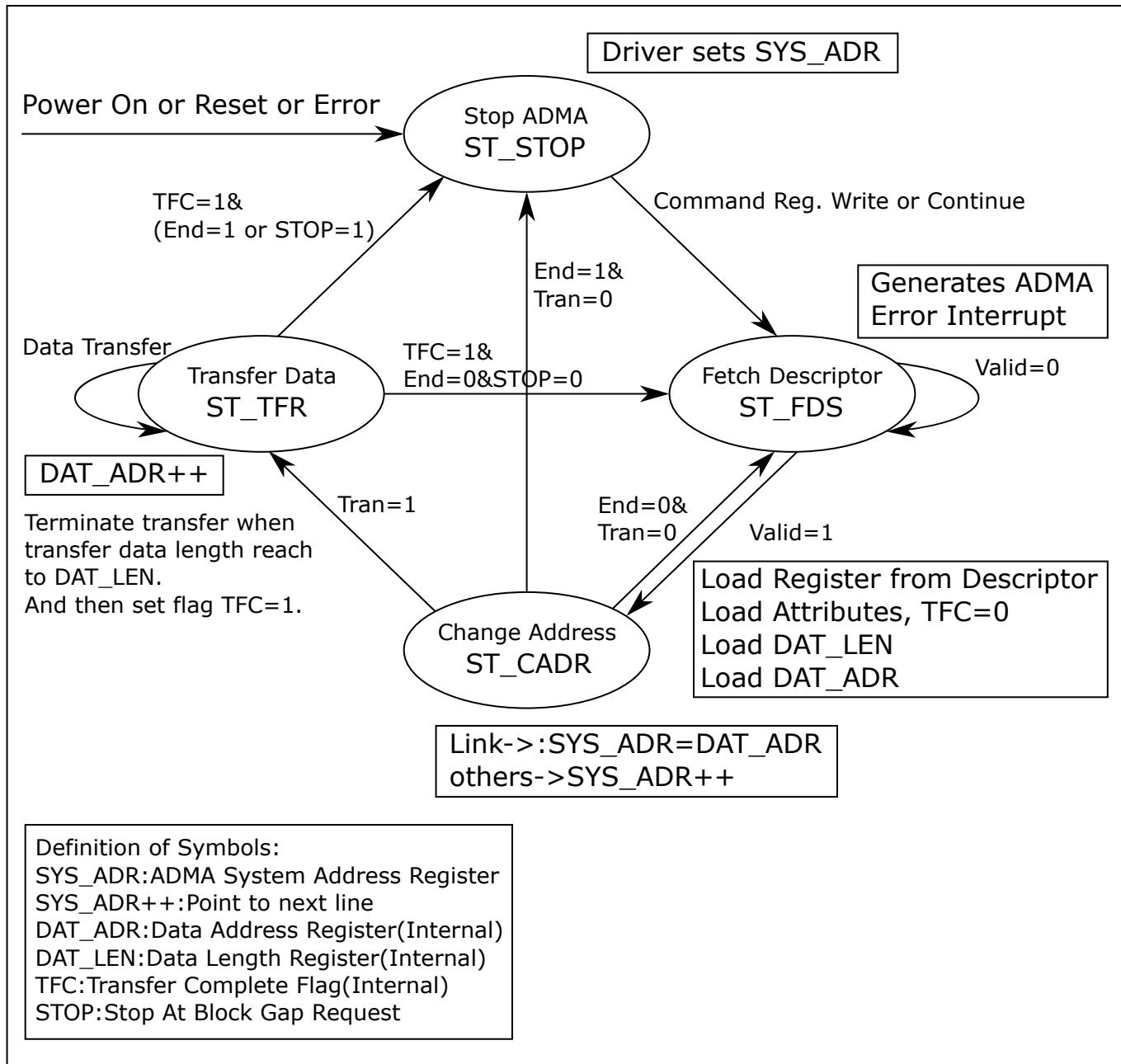


Fig. 23.10: ADMA2 States

- ST_FDS (fetch descriptor): ADMA2 fetches the descriptor line and sets the parameters in the internal register, and then switches to the ST_CADR state.
- ST_CADR (change address): The “Link” operation loads another descriptor address into the ADMA system address register. In other operations, the ADMA system address register is incremented to the next descriptor

line. If End=0, it switches to the ST_TFR state. Even if some errors occur, ADMA2 must not stop in this state.

- ST_TFR (transfer data): The data of one descriptor line is transferred between system memory and SD card. If data transfer continues (End=0), it switches to the ST_FDS state. If data transfer is completed, it switches to the ST_STOP state.
- ST_STOP (stop DMA): ADMA2 will remain in this state in these cases: after power-on reset or software reset; all the descriptor data has been transferred. It switches to the ST_FDS state if a new ADMA2 operation is started by writing to the command register.

ADMA2 does not support the Suspend/Resume function, but can use the Stop and Resume functions. When the block gap stop request in the block gap control register is set during ADMA2 operation, the block gap event interrupt will be generated when ADMA2 stops at the block gap. The host controller shall use the Read Wait or Stop SD Clock to stop the ADMA2 read operation. When ADMA2 is stopped, no SD command can be issued. Errors occurred during ADMA2 transfer may stop ADMA2 operation and generate an ADMA error interrupt. The ADMA Error Status field in the ADMA Error Status register stores the status that ADMA2 has stopped. The host driver can identify the location of the error descriptor in two ways. That is, if ADMA stops in the ST_FDS state, the ADMA system address register will point to the error descriptor line. If ADMA stops in the ST_TFR or ST_STOP state, the ADMA system address register will point to the next line of the error descriptor line. Therefore, ADMA2 must not stop in the ST_CADR state.

23.3.13 Test Register

The test register is defined for the purpose of testing. When it is difficult to intentionally generate some interrupts, you may use this function to manually generate such interrupts for driver debugging. To that end, a force event register is defined to control the Error Interrupt Status and the automatic CMD error status. It is also difficult to control card insertion and removal intentionally. The card detection signal selection and card detection test level in the host control 1 register make it possible to manually control the card inserted in the current status register and generate card insertion and card removal interrupts in the Normal Interrupt Status register.

23.3.14 Block Count

The block count command (CMD23) provides an untimed method to stop multi-block operations. The block count is set in the parameter of CMD23 to specify the transfer length of CMD18 or CMD25 after that. Automatic CMD23 is the function of automatically sending CMD23 before sending CMD18 or CMD25. This function aims to avoid performance degradation during memory access by deleting the interrupt service of CMD23. The offset 008h parameter 1 register is used for CMD18 or CMD25. Then an offset of 000h is assigned to the parameter 2 register of CMD23. The host controller does not use the parameter 2 register to calculate the data transfer length. There are two cases of data length of the host-side data transfer operation: non-ADMA and ADMA. The total length of AMDA descriptor refers to the sum of all the 16-bit data length of each ADMA descriptor line. The “block count enable” shall be disabled for ADMA. It is worth noting that the total data transfer length of the host controller shall be equal to the transfer length of the card.

23.3.15 Sampling Clock Tuning

In UHSI mode, the SD bus can run in the high clock frequency mode, and then the data window of cards on CMD and DAT[3:0] lines becomes smaller. The position of the data window varies with the implementation of the card and host system. Hence, when SDR104 or SDR50 is supported by executing the tuning program and adjusting the sampling clock (if the usage tuning of SDR50 is set to 1 in the capability register), the host controller shall support the tuning circuit. The execution tuning and sampling clock selection in the host control 2 register is used to control the tuning circuit.

23.3.16 SD Host Standard Register

23.3.16.1 SD Host Control Register Mapping

The following table summarizes the standard SD host controller register sets. The host driver needs to determine the base address of the register set by the host system specific method. The size of the register set is 256 bytes. For multiple card slot controllers, one register set is assigned to each card slot, but the register at the offset 0F0h0FFh is assigned as a common area, and these registers contain the same value from each card slot register set.

Offset	15-08 bit	07-00 bit	Offset	15-08 bit	07-00 bit
002h	SDMA System Address(High),Argument 2(High)		000h	SDMA System Address(Low),Argument 2(Low)	
006h	Block Count		004h	Block Size	
00Ah	Argument 1(High)		008h	Argument 1(Low)	
00Eh	Command		00Ch	Transfer Mode	
012h	Response1		010h	Response0	
016h	Response3		014h	Response2	
01Ah	Response5		018h	Response4	
01Eh	Response7		01Ch	Response6	
022h	Buffer Data Port1		020h	Buffer Data Port0	
026h	Present State		024h	Present State	
02Ah	Wakeup Control	Block Gap Control	028h	Power Control	Host Control 1
02Eh	Software Reset	Timeout Control	02Ch	Clock Control	
032h	Error Interrupt Status		030h	Normal Interrupt Status	
036h	Error Interrupt Status Enable		034h	Normal Interrupt Status Enable	
03Ah	Error Interrupt Signal Enable		038h	Normal Interrupt Signal Enable	
03Eh	Host Control 2		03Ch	Auto CMD Error Status	
042h	Capabilities		040h	Capabilities	
046h	Capabilities		044h	Capabilities	
04Ah	Maximum Current Capabilities		048h	Maximum Current Capabilities	
04Eh	Maximum Current Capabilities(Reserved)		04Ch	Maximum Current Capabilities(Reserved)	
052h	Force Event for Error Interrupt Status		050h	Force Event for Auto CMD Error Status	
056h	---		054h	---	ADMA Error Status
05Ah	ADMA System Address [31:16]		058h	ADMA System Address [15:00]	
05Eh	ADMA System Address [63:48]		05Ch	ADMA System Address [47:32]	
062h	Preset Value		060h	Preset Value	
066h	Preset Value		064h	Preset Value	
06Ah	Preset Value		068h	Preset Value	
06Eh	Preset Value		06Ch	Preset Value	
---	---		---	---	
0E2h	Shared Bus Control(High)		0E0h	Shared Bus Control(Low)	
---	---		---	---	
0F2h	---		0F0h	---	
---	---		---	---	
0FEh	Host Controller Version		0FCh	Slot Interrupt Status	

Fig. 23.11: SD Host Control Register Mapping

23.3.16.2 Configuration of Register Type

The configuration register field is assigned an attribute described in the following table:

Register Attribute	Description
RO	Read-only register: Register bits are read-only and cannot be altered by software or any reset operation. Writes to these bits are ignored.
ROC	Read-only status: These bits are initialized to zero at reset. Writes to these bits are ignored.
RW	Read-Write register: Register bits are read-write and may be either set or cleared by software to the desired state.
RW1C	Read-only status, Write-1-to-clear status: Register bits indicate status when read, a set bit indicating a status event may be cleared by writing a 1. Writing a 0 to RW1C bits has no effect.
RWAC	Read-Write, automatic clear register: The Host Driver requests a Host Controller operation by setting the bit. The Host Controllers shall clear the bit automatically when the operation is complete. Writing a 0 to RWAC bits has no effect.
HwInit	Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.
Rsvd	Reserved. These bits are initialized to zero, and writes to them are ignored.
WO	Write-only register. It is not physically implemented register. Rather, it is an address at which registers can be written.

Fig. 23.12: Types of Registers and Register Bit Fields

23.3.16.3 Initial Value of Register

The host controller sets all registers to their initial values at power-on reset, and the default values of all other registers shall be all bits that are set to zero. The values of the capability register and the maximum current capability register depend on the host controller, and the value of the host controller version register also depends on the host controller.

23.3.16.4 Reserved Bits of a Register

“Reserved” means that this bit can be defined for future use, and it is currently set to 0. These bits shall be set to 0.

24.1 Overview

Secure Digital Input and Output (SDIO) is a peripheral interface developed on the basis of the SD memory card interface. The SDIO card that is compatible with the SD memory card aims to provide high-speed data I/O and low-power mobile electronic devices.

24.2 Features

- For portable and stationary applications
- Minimal modification or no modification to the SD physical bus
- Minimal changes to memory driver software
- Extended physical form factor for special applications
- Plug and play
- Multi-functional support, including multiple I/O, combined I/O and memory
- One card supports up to seven I/O functions and one memory.
- Allow the card to interrupt the host

24.3 Functional Description

24.3.1 Definition of SDIO Signal

24.3.1.1 SDIO Card Type

There are two types of SDIO cards, full-speed and low-speed SDIO cards. In the full clock range of 0–25 MHz, the full-speed card supports SPI, 1-bit SD and 4-bit SD transfer modes. The data transfer rate of a full-speed SDIO card exceeds 100 Mb/s (10 MB/s). The low-speed SDIO card only needs SPI and 1-bit SD transfer modes and supports

the full clock range of 0~400 KHz. The low-speed card intends to provide the low-speed I/O capability with minimum hardware. It supports modem, barcode scanner, GPS receiver, and other functions.

24.3.1.2 SDIO Card Mode

Here are three signal modes defined for SD memory cards that also apply to SDIO cards. First, SPI mode, where Pin 8 is used as an interrupt pin, and all other pins and signaling protocols are the same as that specified in the SD Physical Layer Specification. Second, 1-bit SD data transfer mode, where data is only transferred on the DAT[0] pin. Pin 8 is used as an interrupt pin, and all other pins and signaling protocols are the same as that specified in the SD memory specification. Third, 4-bit SD data transfer mode, where data is transferred on all 4 data pins DAT[3:0]. In this mode, the interrupt pin must not be used exclusively, because it is used as a data transfer line. Thus, if the interrupt function is required, special timing is required to provide the interrupt. The 4-bit SD mode provides the fastest possible data transfer, up to 100 Mb/s.

24.3.1.3 Signal Pin

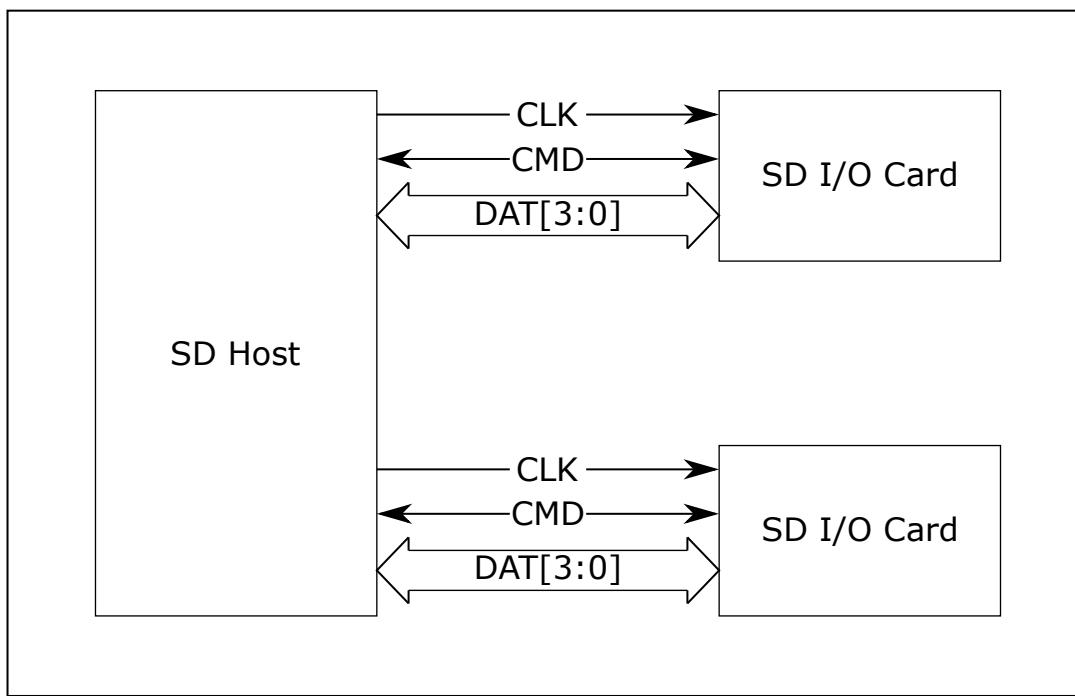


Fig. 24.1: Signal Connection of Two 4-Bit SDIO Cards

24.3.2 SDIO Card Initialization

24.3.2.1 Difference in I/O Card Initialization

The SDIO Specification specifies that when SDIO card is inserted, it shall not cause the failure of non-I/O aware host. To prevent I/O functions from being operated in non-I/O aware hosts, the flow chart of SD card recognition mode shall be changed. A new command (IO_SEND_OP_COND, CMD5) is added to replace ACMD41 and initialize SDIO through the I/O aware host. After reset or power-on, all I/O functions on the card are disabled. When CS is Low, the I/O part of the card will not perform any operation except CMD5 or CMD0. If a SD memory is installed on the card, it shall normally respond to all normal forced memory commands. I/O only cards must not respond to ACMD41, so they are initially displayed as MMC cards. I/O only cards must not respond to the CMD1 used to initialize the MMC card, so they are displayed as non-responding cards. Then the host abandons and disables the card. Therefore, the non-aware host does not receive the response from the I/O only card, and forces it into the inactive state. The following figure shows the operation of the I/O card with non-I/O aware host. The solid line is the actual path, and the dotted line part is not executed:

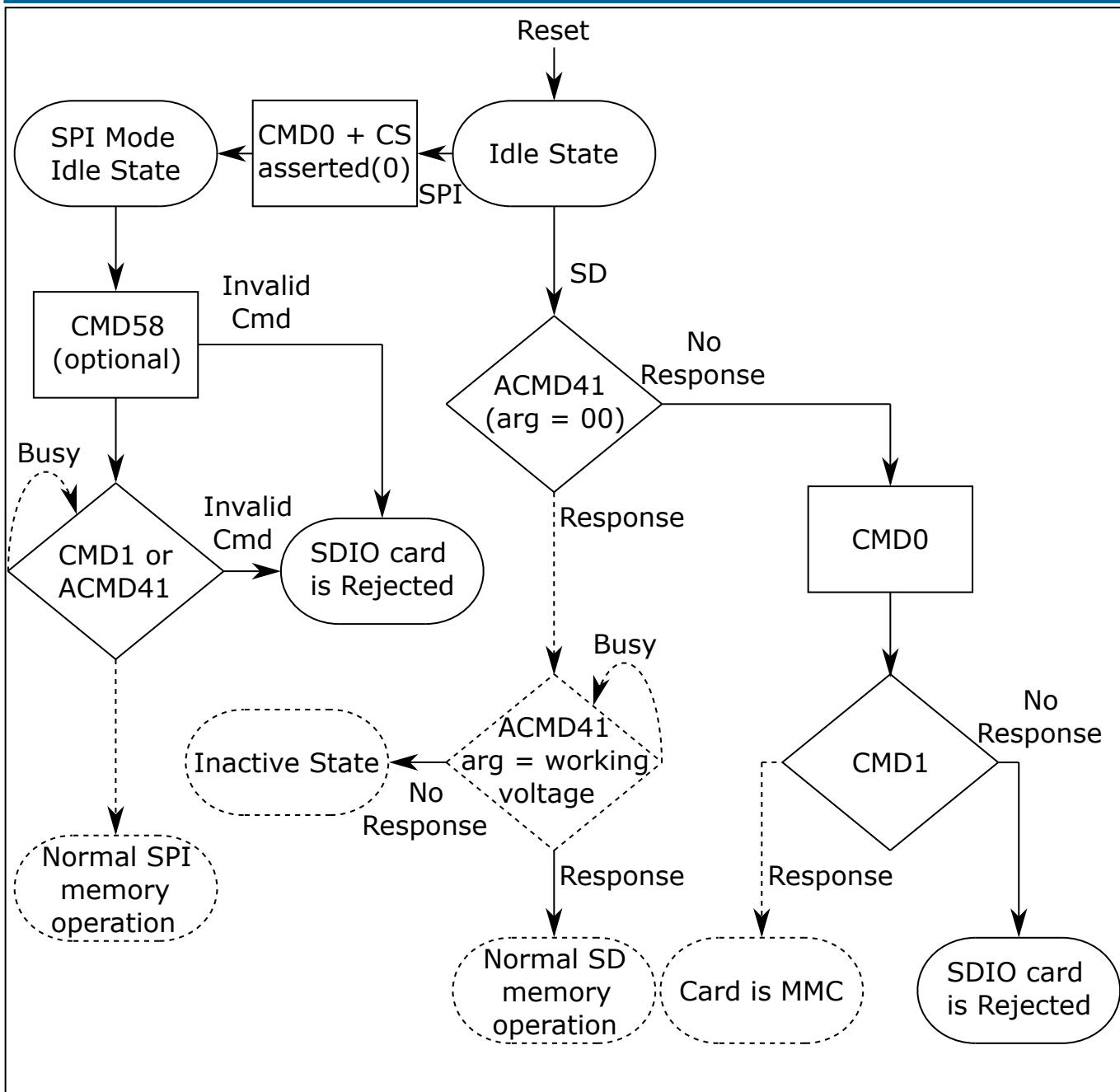


Fig. 24.2: SDIO's Response to Non-I/O Aware Initialization

24.3.2.2 IO_SEND_OP_COND command (CMD5)

The function of the IO_SEND_OP_COND command (CMD5) for the SDIO card is similar to that of ACMD41 for the SD memory card, and it is used to query the required voltage range of the I/O card. The normal response of CMD5 is R4 in SD or SPI format. The format of CMD5 is shown as follows:

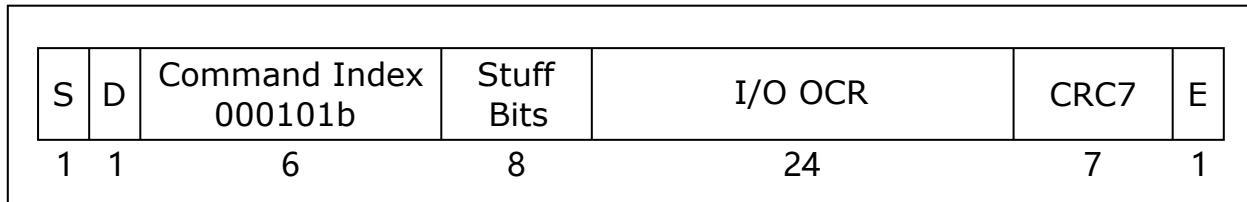


Fig. 24.3: IO_SEND_OP_COND Command

其中：

- S: Start bit, always 0.
- D: Direction, always 1, indicating transfer from host to card.
- Command Index: Identifies the CMD5 command with a value of 000101b.
- Stuff Bits: Not used, shall be set to 0.
- I/O OCR: Operation Conditions Register. The supported minimum and maximum values for VDD.
- CRC7: 7 bits of CRC data.
- E: End bit, always 1.

24.3.2.3 IO_SEND_OP_COND Response (R4)

An SDIO card receiving CMD5 shall respond with a SDIO unique response, R4. The format of R4 for both the SD and SPI modes is:

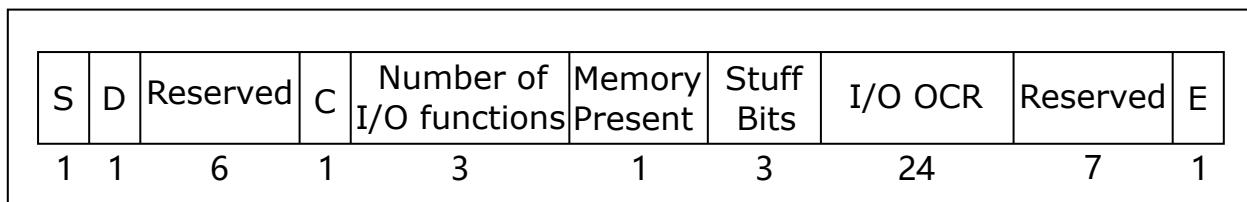


Fig. 24.4: Response R4 in SD Mode

Modified R1	C	Number of I/O functions	Memory Present	Stuff Bits	I/O OCR
8	1	3	1	3	24

Fig. 24.5: Response R4 in SPI Mode

- S: Start bit, always 0.
- D: Direction, always 0, indicating the transfer from card to host.
- Reserved: Bits reserved for future use. These bits shall be set to 1.
- C: Set to 1 if the card is ready to operate after initialization.
- Stuff Bits: Not used, shall be set to 0.
- I/O OCR: Operation Conditions Register. The supported minimum and maximum values for VDD.
- E: End bit, always 1.
- Memory Present: Set to 1 if the card also contains SD memory. Set to 0 if the card is I/O only.
- Number of I/O Functions: Indicates the total number of I/O functions supported by this card. The range is 0-7. Note that the common area present on all I/O cards at function 0 is not included in this count. The I/O functions shall be implemented sequentially beginning at function 1.
- Modified R1: The SPI R1 response byte as described in the SD Physical Layer Specification is modified for I/O as follows:

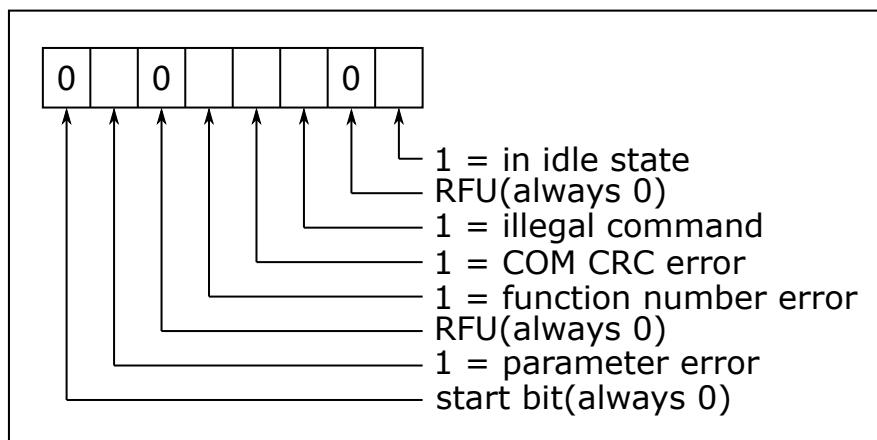


Fig. 24.6: Modified R1 Response

24.3.2.4 Special Initialization Considerations for combo Cards

The host shall be aware of some special situations when initializing a combo card (SDIO plus SD memory on the same card). This is because an implementation of the combo card could actually use two separate controllers (Memory and I/O) in the same package and share the same bus lines. It is important for the host to both detect and properly configure both parts (controllers) of a combo card in order to prevent conflicts between the SDIO and the SD memory controller. These concerns are due to the different responses to a reset (hard or soft) by the two controllers. Another issue is the value of the RCA (Relative Card Address) that exists within the Memory controller. Note that this consideration is for SD 1-bit and SD 4-bit modes only. In SPI mode, card select/de-select is accomplished using the hardware CS line rather than the RCA.

24.3.3 Differences with SD Memory Specification

24.3.3.1 SDIO Command List

The following figure shows the list of commands accepted by SD memory and SDIO cards when using the SD bus interface.

Supported Commands	Abbreviation	SDMEM System	SDIO System	Comments
CMD0	GO_IDLE_STATE	Mandatory	Mandatory	Used to change from SD to SPI mode
CMD2	ALL_SEND_CID	Mandatory		CID not supported by SDIO
CMD3	SEND_RELATIVE_ADDR	Mandatory	Mandatory	
CMD4	SET_DSR	Optional		DSR not supported by SDIO
CMD5	IO_SEND_OP_COND		Mandatory	
CMD6	SWITCH_FUNC	Mandatory	Mandatory	
CMD7	SELECT/DESELECT_CARD	Mandatory	Mandatory	
CMD9	SEND_CSD	Mandatory		CSD not supported by SDIO
CMD10	SEND_CID	Mandatory		CID not supported by SDIO
CMD12	STOP_TRANSMISSION	Mandatory		
CMD13	SEND_STATUS	Mandatory		Card Status includes only SDMEM information
CMD15	GO_INACTIVE_STATE	Mandatory	Mandatory	
CMD16	SET_BLOCKLEN	Mandatory		
CMD17	READ_SINGLE_BLOCK	Mandatory		
CMD18	READ_MULTIPLE_BLOCK	Mandatory		
CMD24	WRITE_BLOCK	Mandatory		
CMD25	WRITE_MULTIPLE_BLOCK	Mandatory		
CMD27	PROGRAM_CSD	Mandatory		CSD not supported by SDIO
CMD28	SET_WRITE_PROT	Optional		
CMD29	CLR_WRITE_PROT	Optional		
CMD30	SEND_WRITE_PROT	Optional		
CMD32	ERASE_WR_BLK_START	Mandatory		
CMD33	ERASE_WR_BLK_END	Mandatory		
CMD38	ERASE	Mandatory		
CMD42	LOCK_UNLOCK	Optional		
CMD52	IO_RW_DIRECT		Mandatory	
CMD53	IO_RW_EXTENDED		Mandatory	Block mode is optional
CMD55	APP_CMD	Mandatory		
CMD56	GEN_CMD	Mandatory		
ACMD6	SET_BUS_WIDTH	Mandatory		
ACMD13	SD_STATUS	Mandatory		
ACMD22	SEND_NUM_WR_BLOCKS	Mandatory		
ACMD23	SET_WR_BLK_ERASE_COUNT	Mandatory		
ACMD41	SD_APP_OP_COND	Mandatory		
ACMD42	SET_CLR_CARD_DETECT	Mandatory		
ACMD51	SEND_SCR	Mandatory		SCR not supported by SDIO

Fig. 24.7: Command List for SD Mode

The following figure shows the list of commands accepted by SD memory and SDIO cards when using the SPI bus interface.

Supported Commands	Abbreviation	SDMEM System	SDIO System	Comments
CMD0	GO_IDLE_STATE	Mandatory	Mandatory	Used to change from SD to SPI mode
CMD1	SEND_OP_COND	Mandatory		
CMD5	IO_SEND_OP_COND		Mandatory	
CMD6	SWITCH_FUNC	Mandatory	Mandatory	
CMD9	SEND_CSD	Mandatory		CSD not supported by SDIO
CMD10	SEND_CID	Mandatory		CID not supported by SDIO
CMD12	STOP_TRANSMISSION	Mandatory		
CMD13	SEND_STATUS	Mandatory		Card Status includes only SDMEM information
CMD16	SET_BLOCKLEN	Mandatory		
CMD17	READ_SINGLE_BLOCK	Mandatory		
CMD18	READ_MULTIPLE_BLOCK	Mandatory		
CMD24	WRITE_BLOCK	Mandatory		
CMD25	WRITE_MULTIPLE_BLOCK	Mandatory		
CMD27	PROGRAM_CSD	Mandatory		CSD not supported by SDIO
CMD28	SET_WRITE_PROT	Optional		
CMD29	CLR_WRITE_PROT	Optional		
CMD30	SEND_WRITE_PROT	Optional		
CMD32	ERASE_WR_BLK_START	Mandatory		
CMD33	ERASE_WR_BLK_END	Mandatory		
CMD38	ERASE	Mandatory		
CMD42	LOCK_UNLOCK	Optional		
CMD52	IO_RW_DIRECT		Mandatory	
CMD53	IO_RW_EXTENDED		Mandatory	Block mode is optional
CMD55	APP_CMD	Mandatory		
CMD56	GEN_CMD	Mandatory		
CMD58	READ_OCR	Mandatory		
CMD59	CRC_ON_OFF	Mandatory	Mandatory	
ACMD13	SD_STATUS	Mandatory		
ACMD22	SEND_NUM_WR_BLOCKS	Mandatory		
ACMD23	SET_WR_BLK_ERASE_COUNT	Mandatory		
ACMD41	SD_APP_OP_COND	Mandatory		
ACMD42	SET_CLR_CARD_DETECT	Mandatory		
ACMD51	SEND_SCR	Mandatory		SCR includes only SDMEM information

Fig. 24.8: Command List for SPI Mode

24.3.3.2 Unsupported SD Memory Commands

Several commands required for SD memory cards are not supported by either SDIO-only cards or the I/O portion of combo cards. Some of these commands have no use in SDIO cards such as Erase commands and thus are not supported in SDIO. Moreover, there are several commands for SD memory cards that have different commands when used with the SDIO section of a card. The following table lists these SD memory commands and the equivalent SDIO commands.

SD Memory Command	SDIO Command	Comments
CMD0	CMD52 (写入CCCR中的I/O复位)	重置命令 (CMD0) 仅用于组合卡的内存或内存部分。要重置仅I/O卡或组合卡的I/O部分, 请使用CMD52将a1写入CCCR中的RES位 (寄存器6的位3)。请注意, 在SD模式下, CMD0仅用于指示进入SPI模式, 应予以支持。仅I/O卡或组合卡的I/O部分不会使用CMD0重置
CMD12	CMD52 (写入I/O中止)	为了中止数据块传输, SD内存使用CMD12。要中止I/O事务, 请使用CMD52写入CCCR中的中止寄存器 (寄存器6的位2:0)
CMD16	CMD52 (写入I/O块长度)	CMD16设置SD内存的块长度。要为每个I/O函数设置块长度, 请使用CMD52在FBR中写入块长度
CMD2	无	仅SDIO卡中不存在CID寄存器
CMD4	无	仅SDIO卡中不存在DSR寄存器
CMD9	无	仅SDIO卡中不存在CSD寄存器
CMD10	无	仅SDIO卡中不存在CID寄存器
CMD13	无	仅SDIO卡或组合卡的I/O部分不支持SD内存使用的相同的SEND_STATUS (CMD13) 协议
ACMD6	CMD52 (写入CCCR中的Bus_Width[1:0])	SET_BUS_WIDTH由对CCCR的写入处理
ACMD13	无	仅SDIO卡中不存在SD状态寄存器
ACMD41	CMD5	SDIO卡和主机使用IO_SEND_OP_COND命令 (CMD5)
ACMD42	CMD52	在SD模式下, DAT[3]上的上拉电阻通过写入CCCR中的CD禁用位来控制。对于组合卡, 此电阻启用, 除非内存和I/O控制寄存器都设置为禁用电阻
ACMD51	无	仅SDIO卡中不存在SCR寄存器
CMD17 CMD18 CMD24 CMD25	CMD53	I/O块操作使用CMD53, 而不是内存块读/写命令

Fig. 24.9: Unsupported SD Memory Commands

24.3.3.3 Modified R6 Response

The normal response to CMD3 by a memory card is R6, as shown in the following table:

Bit position	47	46	[45:40]	[39:8]Argument field	[7:1]	0	
Width(bits)	1	1	6	16	16	7	1
Value	'0'	'0'	X	X	X	X	'1'
Description	Start bit	Direction bit	Command index ('000011')	New published RCA [31:16] of the card	[15:0] Card status	CRC7	end bit

Fig. 24.10: R6 Response of CMD3

The card status bits (8–23) are changed when CMD3 is sent to an I/O only card. In this case, the 16 bits of response shall be the SDIO-only values shown in the following table.

Bits	Identifier	Type	Value	Description	Clear Condition
15	COM_CRC_ERROR	E R	'0' = no error '1' = error	上一个命令的CRC检查失败	B
14	ILLEGAL_COMMAND	E R	'0' = no error '1' = error	命令对于卡状态不合法	B
13	ERROR	E R X	'0' = no error '1' = error	操作过程中发生常规错误或未知错误	C
12:0	未定义。仅SDIO卡的读数应为0。主机应该忽略这些位				

Fig. 24.11: SDIO R6 Status Bit

24.3.3.4 Reset for SDIO

In order to reset all functions within an SDIO card or the SDIO portion of a combo card, a method different than that used for SD memory is defined. The reset command (CMD0) is only used for memory or the memory portion of combo cards. In order to reset an I/O only card or the I/O portion of a combo card, use CMD52 to write a 1 to the RES bit in the CCCR (bit 3 of register 6). Note that in the SD mode, CMD0 is only used to indicate entry into SPI mode. An I/O only card or the I/O portion of a combo card is not reset by CMD0.

24.3.3.5 Bus Width

For an SD memory card, the bus width for SD mode is set using ACMD6. For an SDIO card, a write to the CCCR using CMD52 is used to select bus width. In the case of a combo card, both selection methods exist. In this case, the host shall set the bus width in both locations by issuing both the ACMD6 and the CCCR write using CMD52 with the same width before starting any data transfer.

24.3.3.6 Card Detect Resistor

SD memory and I/O cards use a pull-up resistor on DAT[3] to detect card insertion. The procedure to enable/disable this resistor is different between SD memory and SDIO. SD memory uses ACMD42 to control this resistor while SDIO uses writes to the CCCR using CMD52. In the case of a combo card, both control modes exist and shall be managed by the host. For a combo card, the resistor is enabled only when both the memory and the I/O control registers have the resistor enabled. That is, after a power on, the host shall disable the resistor by sending ACMD42 to the memory controller or a CCCR write to the SDIO controller since the resistor enable is a logical AND of the two enables. After power-up, both locations default to resistor enabled. It is worth noting that after an I/O reset, the I/O resistor enable is not changed.

24.3.3.7 Data Transfer Block Sizes

SDIO cards may transfer data in either a multi-byte (1 to 512 bytes) or an optional block format, while the SD memory cards are fixed in the block transfer mode. The SD Physical Layer Specification limits the block size for data transfer to powers of 2 (i.e. 512, 1024, 2048) unless using partial read and write. The SDIO Specification allows any block size from 1 byte to 2048 bytes in order to accommodate the various natural block sizes for I/O functions. It is worth noting that an SDIO card function may define a maximum block size or byte count in the CIS that is smaller than the maximum values described above.

24.3.3.8 Data Transfer Abort

A host communicating with a SD memory card uses CMD12 to abort the transfer of read or write data from/to the card. For an SDIO card, CMD12 abort is replaced by a write to the ASx bits in the CCCR. Normally, the abort is used to stop an infinite block transfer (block count=0). If an exact number of blocks are to be transferred, it is recommended that the host issue a block command with the correct block count, rather than using an infinite count and aborting the data at the correct time.

24.3.4 New I/O Read/Write Commands

Two additional data transfer instructions have been added to support I/O. IO_RW_DIRECT is a direct I/O command similar to MMC's "fast I/O" command. IO_RW_EXTENDED allows fast access with byte or block addresses. Both commands are in class 9 (I/O Commands).

24.3.4.1 IO_RW_DIRECT Command (CMD52)

The IO_RW_DIRECT is the simplest means to access a single register within the 128K of register space in any I/O function, including the common I/O area (CIA). This command reads or writes 1 byte using only 1 command/response pair. A common use is to initialize registers or monitor status values for I/O functions. This command is the fastest means to read or write single I/O registers, as it requires only a single command/response pair. The command structure is shown as follows:

S	D	Command Index 110100b	R/W flag	Function Number	RAW flag	Stuff	Register Address	Stuff	Write Data or Stuff Bits	CRC7	E
1	1	6	1	3	1	1	17	1	8	7	1

Fig. 24.12: IO_RW_DIRECT Command

- S: Start bit, always 0.
- D: Direction, always 1, indicating transfer from host to card.
- Command Index: Identifies the "IO_RW_DIRECT" command with a value of 110100b.
- R/W Flag: This bit determines the direction of the I/O operation. If this bit is 0, this command shall read data from the SDIO card at the address specified by the Function Number and the Register Address to the host. The data byte is returned in the response, R5. If this bit is set to 1, the command shall write the bytes in the Write Data field to the I/O location addressed by the Function Number and the Register Address. If the RAW flag is 0, then the data in the register that was written shall be read and that value returned in the response.
- RAW Flag: The Read after Write flag. If this bit is set to 1 and the R/W flag is set to 1, then the command shall read the value of the register after the write. This is useful for allowing writing to the control register and reading

the state of the same address. If this bit is cleared, the value returned in the R5 response shall be the same as the write data in the command. If this bit is set, the data field of the R5 response shall contain the value read from the addressed register after the write operation.

- Function Number: The number of the function within the I/O card you wish to read or write. Note that function 0 selects the common I/O area (CIA).
- Register Address: This is the address of the byte of data inside of the selected function to read or write. There are 17 bits of address available so the register is located within the first 128K (131,072) addresses of that function.
- Write Data/Stuff Bits: For a direct write command (R/W=1), this is the byte that is written to the selected address. For a direct read (R/W=0), this field is not used and shall be set to 0.
- CRC7: 7 bits of CRC data.
- E: End bit, always 1.

24.3.4.2 IO_RW_DIRECT Response (R5)

The SDIO card's response to CMD52 shall be in one of two formats. If the communication between card and host is in the 1-bit or 4-bit SD mode, the response shall be in a 48-bit response (R5). If the operation was a read command, the data being read is returned as an 8-bit value. In addition, 15 bits of status information is returned. The format of the response is as follows:

S	D	Command Index 110100b	Stuff 16	Response Flags 7----- Bit ----- 0 8	Read or Write Data 8	CRC7 7	E 1
1	1	6					

Fig. 24.13: IO_RW_DIRECT Response in SD Mode

- S: Start bit, always 0.
- D: Direction, always 0, indicating the transfer from card to host.
- Command Index: Identifies the “IO_RW_DIRECT” command with a value of 110100b.
- Stuff Bits: Not used, shall be set to 0.
- Response Flags: 8 bits of flag data indicating the status of the SDIO card.
- Read or Write Data: For an I/O write (R/W=1) with the RAW Flag set (RAW=1), this field shall contain the value read from the addressed register after the write of the data contained in the command. Note that in this case, the read-back data may not be the same as the data written to the register, depending on the design of the hardware. For an I/O write with the RAW bit=0, the SDIO function shall not do a read after write operation, and the data in this field shall be identical to the data byte in the write command. For an I/O read (R/W=0), the actual value read

from that I/O location is returned in this field.

- CRC7: 7 bits of CRC data.
- E: End bit, always 1.

If the communication is using the SPI mode, the response shall be a 16-bit R5 response. If the operation was a read command, the data being read is returned as an 8-bit value. In addition, 8 bits of status information is returned in a SPI R1 response byte. The format of the response is as follows:

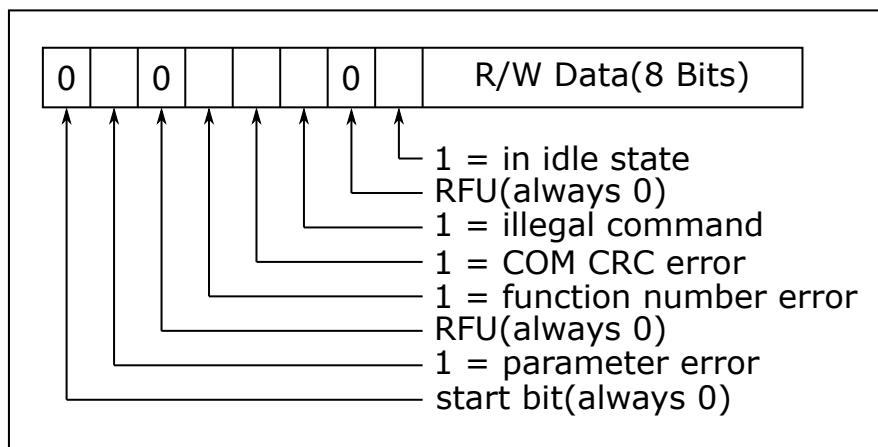


Fig. 24.14: IO_RW_DIRECT Response in SPI Mode

Note: The read/write (R/W) data is identical to the read/write data described for the SD R5 response. Parameter error status in SPI mode corresponds to OUT_OF_RANGE and ERROR in the SD mode response. In the case of CMD53, Data Error Token shall also be used to indicate OUT_OF_RANGE and ERROR.

24.3.4.3 IO_RW_EXTENDED Command (CMD53)

In order to read and write multiple I/O registers with a single command, a new command, IO_RW_EXTENDED is defined. This command is included in command class 9 (I/O Commands). This command allows the reading or writing of a large number of I/O registers with a single command. Since this is a data transfer command, it provides the highest possible transfer rate. The IO_RW_EXTENDED command is shown as follows:

S	D	Command Index 110101b	R/W flag	Function Number	Block Mode	OP Code	Register Address	Byte/Block Count	CRC7	E
1	1	6	1	3	1	1	17	9	7	1

Fig. 24.15: IO_RW_EXTENDED Command

- S: Start bit, always 0.
- D: Direction, always 1, indicating transfer from host to card.

- Command Index: Identifies the “IO_RW_EXTENDED” command with a value of 110101b.
- R/W Flag: This bit determines the direction of the I/O operation. If this bit is 0, this command reads data from the SDIO card at the address specified by the Function Number and the Register Address to the host. The read data shall be returned on the DAT[x] lines. If this bit is set to 1, the command shall write the bytes from the DAT[x] lines to the I/O location addressed by the Function Number and the Register Address.
- Function Number: The number of the function within the I/O card you wish to read or write. Note that function 0 selects the common I/O area (CIA).
- Block Mode: (Optional) this bit, if set to 1, indicates that the read or write operation shall be performed on a block basis, rather than the normal byte basis. If this bit is set, the byte/block count value shall contain the number of blocks to be read/written. The block size for functions 1-7 is set by writing the block size to the I/O block size register in the FBR. The block size for function 0 is set by writing to the FN0 block size register in the CCCR. Card and host support of the block I/O mode is optional. The host can determine if a card supports block I/O by reading the Card supports MBIO bit (SMB) in the CCCR. The block size used when Block Mode = 1 and the maximum byte count per command used when Block Mode = 0 can be read from the CIS in the tuple TPLFE_MAX_BLK_SIZE on a per-function basis.
- OP code: Defines the read/write operation. 0 is used to read or write multiple bytes of data to/from a fixed address. 1 is used to read or write multiple bytes of data to/from an incremental address.
- Register Address: Start Address of I/O register to read or write. Range is [0~0x1FFF].
- Byte/Block Count: If the command is operating on bytes (Block Mode = 0), this field contains the number of bytes to read or write. A value of 0X000 shall cause 512 bytes to be read or written.
- CRC7: 7 bits of CRC data.
- E: End bit, always 1.

24.3.5 SDIO Card Internal Operation

I/O access differs from memory in that the registers can be written and read individually and directly without a FAT file structure or the concept of blocks (although block access is supported). These registers allow access to the I/O data, control of the I/O function, report on status or transfer I/O data to the host. SD memory relies on the concept of a fixed block length with commands reading/writing multiples of these fixed size blocks. I/O may or may not have fixed block lengths and the read size may be different from the write size. Because of this, I/O operations may be based on either a length (byte count) or a block size.

24.3.5.1 Overview

Each SDIO card may have from 1 to 7 functions plus one built-in memory function. A function is a self-contained I/O device. I/O functions may be identical or completely different from each other. All I/O functions are organized as a collection of registers. There is a maximum of 131,072 (2^{17}) registers possible for each I/O function. These registers and their individual bits may be Read Only (RO), Write Only (WO) or Read/Write (R/W). These registers can be 8, 16 or 32 bits wide within the card. All addressing is based on byte access. These registers can be written and/or read one at a time, multiply to the same address or multiply to an incremental address. The single R/W access is often used to initialize the I/O function or to read a single status or data value. The multiple reads to a fixed address are used to read or write data from a data FIFO register in the card. The read to incremental addresses is used to read or write a collection of data to/from a RAM area inside of the card. The following figure shows the mapping of the CIA and optional CSA space for an SDIO card.

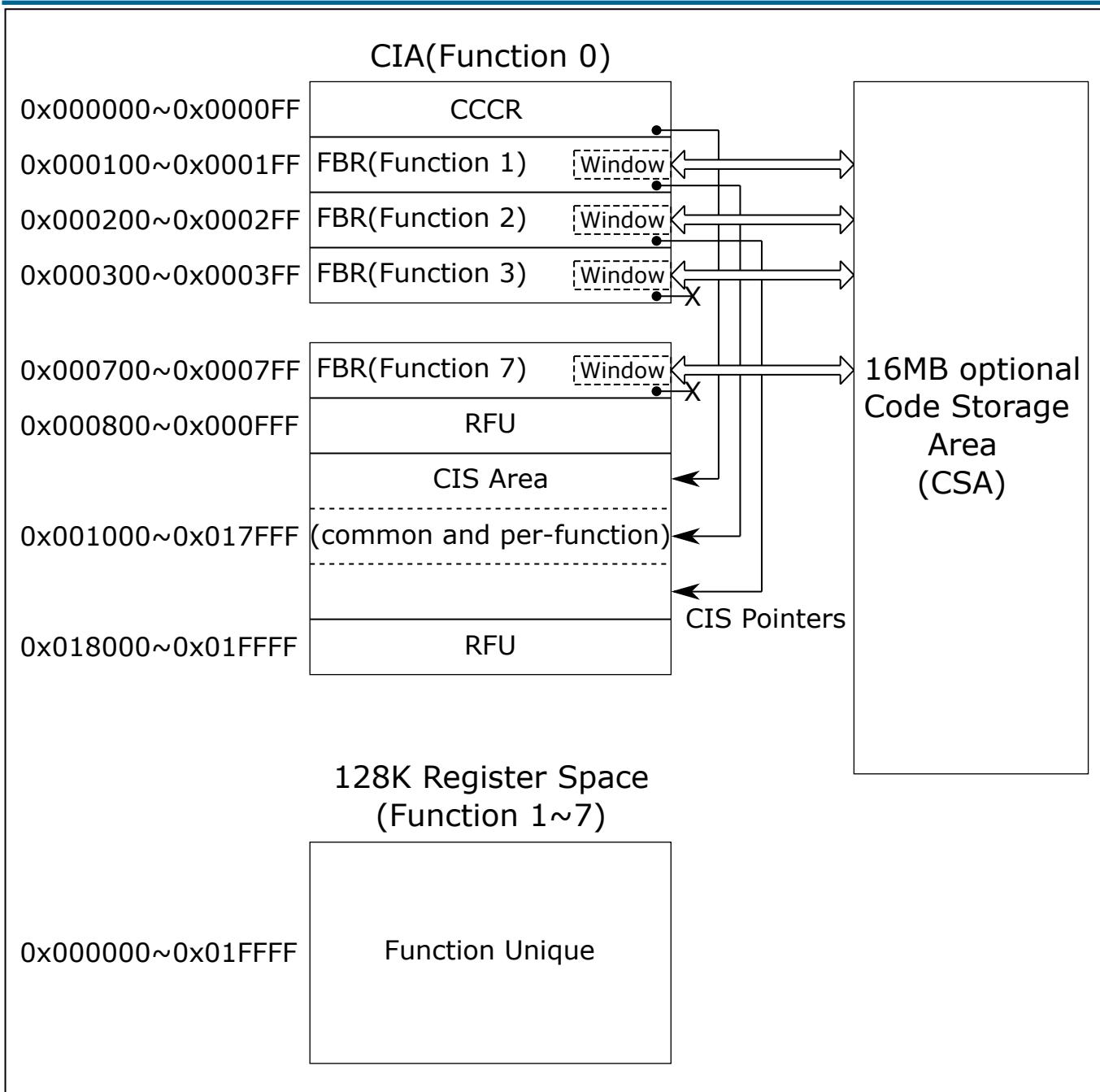


Fig. 24.16: SDIO Internal Mapping

24.3.5.2 Register Access Time

All registers in SDIO only cards and the SDIO portion of combo cards shall complete read and write data transfers in less than one second. This timeout value relates to the time for the requested data to be transferred to/from the host on the DAT[X] lines and not the timing between the command and the response. This wait time is signaled to the host by the card using “busy” for a write or delaying the start bit for a read operation. The host can use one second as the timeout value for a non-responding location.

24.3.5.3 Interrupt

All SDIO hosts shall support hardware interrupts. If a host does not support interrupts, it may have difficulties working with SDIO cards that expect fast response to interrupt conditions. Each function within an SDIO or combo card may implement interrupts as needed. The interrupt used on SDIO functions is a type commonly called “level sensitive”. Level sensitive means that any function may signal for an interrupt at any time, but once the function has signaled an interrupt, it shall not release (stop signaling) the interrupt until the cause of the interrupt is removed or commanded to do so by the host. Since there is only one interrupt line, it may be shared by multiple interrupt sources. The function shall continue to signal the interrupt until the host responds and clears the interrupt. Since multiple interrupts may be active at once, it is the responsibility of the host to determine the interrupt source(s) and deal with it as needed. This is done on the SDIO function by the use of two bits, the interrupt enable and interrupt suspend. Each function that may generate an interrupt has an interrupt enable bit. In addition, the SDIO card has a master interrupt enable that controls all functions. An interrupt shall only be signaled to the SD bus if both the function’s enable and the card’s master enable are set. The second interrupt bit is called interrupt suspend. This read-only bit tells the host which function(s) may be signaling for an interrupt. There is an interrupt suspend bit for each function that can generate interrupts. These bits are located in the CCCR area.

24.3.5.4 Suspend/Resume

Within a multi-function SDIO or a combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of Suspend/Resume. If a card supports Suspend/Resume, the host may temporarily halt a data transfer operation to one function or memory (suspend) in order to free the bus for a higher priority transfer to a different function or memory. Once this higher-priority transfer is completed, the original transfer is re-started where it left off (resume). Support of Suspend/Resume is optional on a per-card basis. If Suspend/Resume is implemented, it shall be supported by the memory of a combo card and all I/O functions except 0 (the CIA). It is worth noting that the host can suspend multiple transactions and resume them in any order desired. The I/O function 0 does not support Suspend/Resume. Any card that supports Suspend/Resume shall also support Read Wait and Direct Commands (SRW and SDC = 1). It is worth noting that Suspend/Resume is defined only for the SD 1 and 4-bit modes. It does not apply to SPI transfers.

24.3.5.5 Read Wait

Host devices built based on the SD Physical Layer Specification shall control the SDCLK to stop the read data block output from a card executing a multiple read command whenever the host cannot accept more data. During the time that the host has stopped the SDCLK, a CMD52 cannot be issued. This limitation causes a problem in that a host device built based on the SD Physical Layer Specification cannot perform the I/O command during a multiple read cycle. In order to eliminate this limitation, the SDIO Specification adds the Read Wait control to enable the host to issue CMD52 during a multiple read cycle. Read Wait uses the DAT[2] line to allow the host to signal the card to temporarily halt the sending of read data by a card. This feature is optional for an SDIO or combo card. However, if

an SDIO or combo supports Read Wait, all functions and any memory shall support Read Wait. Any card that supports Suspend/Resume shall also support Read Wait. It is worth noting that Read Wait is defined only for the SD 1 and 4-bit modes. It does not apply to SPI transfers.

24.3.5.6 CMD52 During Data Transfer

A card may accept CMD52 during data transfer if it supports Direct Commands. For both SD and SPI modes, if an error occurs during data transfer the SDIO card shall accept CMD52 to allow I/O abort and reset regardless of this bit value of the value of SDC.

24.3.5.7 Fixed Internal Mapping

The SDIO card has a fixed internal register space and a function unique area. The fixed area contains information about the card and certain mandatory and optional registers in fixed locations. The fixed locations allow any host to obtain information about the card and perform simple operations such as Enable in a common manner. The function unique area is a per-function area, which is defined either by the Application Specifications for Standard SDIO functions or by the vendor for non-standard functions.

24.3.5.8 Common I/O Area (CIA)

The CIA shall be implemented on all SDIO cards. The CIA is accessed by the host via I/O reads and writes to function 0. The registers within the CIA are provided to enable/disable the operation of the I/O function(s), control the generation of interrupts and optionally load software to support the I/O functions. The registers in the CIA also provide information about the function(s) abilities and requirements. There are three distinct register structures supported within the CIA. They are:

- Card Common Control Registers (CCCR)
- Function Basic Registers (FBR)
- Card Information Structure (CIS)

24.3.5.9 Card Common Control Registers (CCCR)

CCCR allows for quick host checking and control of an I/O card's enable and interrupts on a per card (master) and per function basis. The bits in the CCCR are mixed Read/Write and read only. If any of the possible 7 functions are not provided on an SDIO card, the bits corresponding to unused functions shall all be read-only and read as 0. All reserved for future use bits (RFU) shall be read-only and return a value of 0. All writeable bits are set to 0 after power-up or reset. Access to the CCCR is possible even after initialization when the I/O functions are disabled. This allows the host to enable functions after initialization.

24.3.5.10 Function Basic Registers (FBR)

In addition to the CCCR, each supported I/O function has a 256-byte area used to allow the host to quickly determine the abilities and requirements of each function, enable power selection for each function and to enable software loading. The address of this area is from 0x00n00 to 0x00nFF where n is the function number (0x1 to 0x7).

24.3.5.11 Card Information Structure (CIS)

The Card Information Structure provides more complete information about the card and the individual functions. The CIS is the common area to read information about all I/O functions that exist in a card. The design is based on the PC Card16 design standardized by PCMCIA. All cards that support I/O shall have a common CIS and a CIS for each function. The CIS is accessed by reading the 0x1000 to 0x17FFF area. This area serves the card as a Common CIS and also as the storage area for each function. The common area and each function have a pointer to the start of its CIS within this memory space.

24.3.5.12 Multiple Function SDIO Cards

Multiple Function SDIO Cards shall have a separate set of configuration registers for each function on the card. Multiple Function SDIO Cards shall use a combination of a CIS common to all functions on the card and a separate function-specific CIS specific to each function on the card. The common CIS describes features that are common to all functions on the card. Each function-specific CIS describes features specific to a particular function on the SDIO Card. Functions are numbered sequentially beginning with 1. The CMD5 response indicates the total number of functions, which includes ‘dummy’ functions. The host shall iterate through the CIS entries based on the CMD5 response. The ERROR status flag of an R5 response is type “E R X”, and can indicate an error in the previous command. Since the host software needs a method to determine which function detected the error, a Multiple Function SDIO Card shall only return the R5 ERROR status flag in the subsequent command issued to the same function.

24.3.5.13 Setting Block Size with CMD53

The host sets the block size for a function's multiple block transfers by writing to the 16-bit function I/O block size register in the FBR. The host shall not write this register using CMD53 with Block Mode set to 1. If the card detects an invalid block size before executing CMD53 with Block Mode set to 1, it shall indicate an OUT_OF_RANGE error in the current response and shall not perform data transfer. This will also stop the interrupt cycle

24.3.5.14 Bus State Diagram

The following figure shows the Bus State Diagram for an SDIO card. It shows the bus states and their relations to SDIO commands and Suspend/Resume.

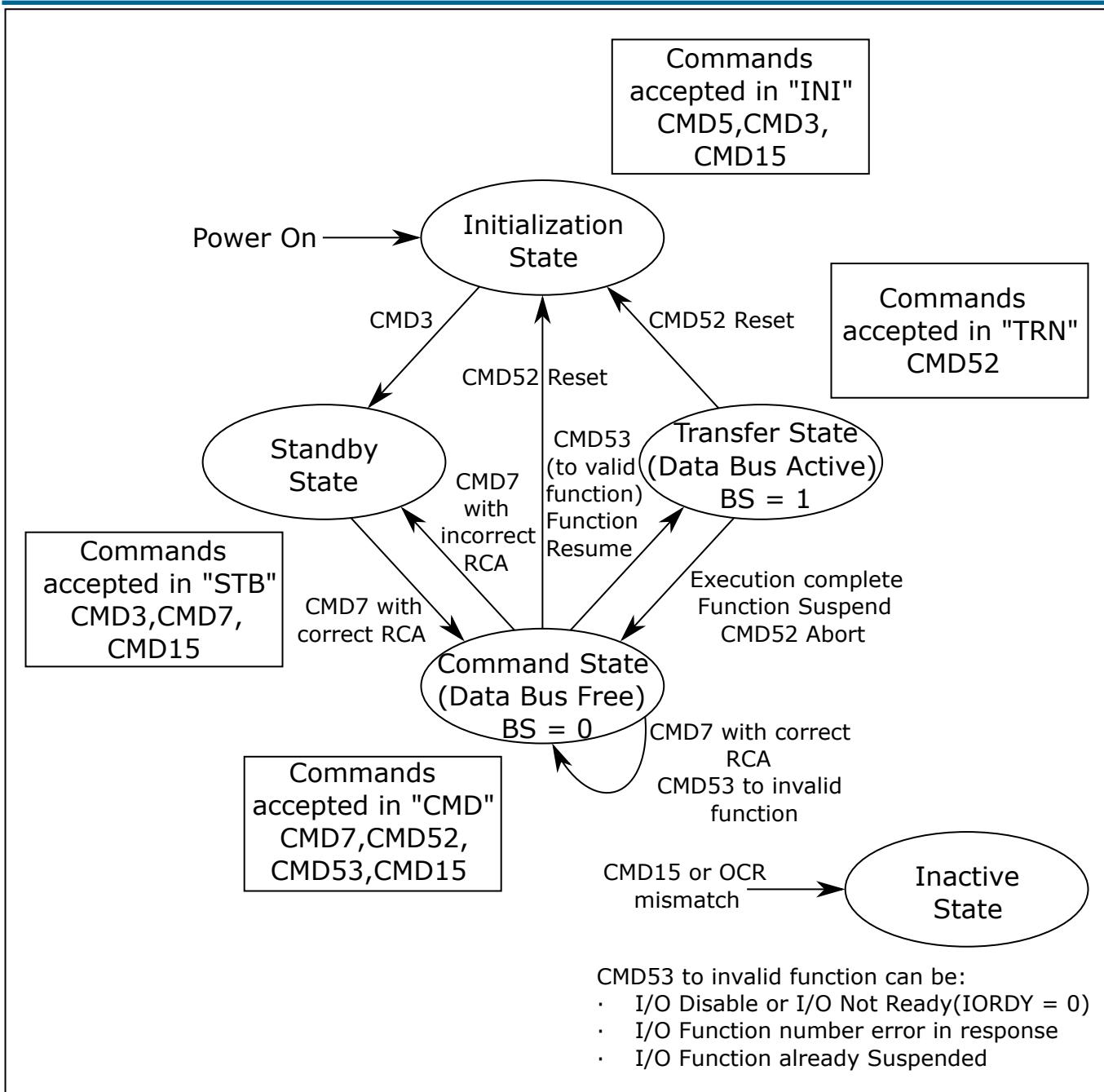


Fig. 24.17: State Diagram for Bus State Machine

24.3.6 Embedded I/O Code Storage Area (CSA)

In order to support the concept of “Plug-and-Play” for SDIO cards, each function contained in a card may need to contain a block of memory for the storage of drivers and/or applications. In addition, since the same SDIO card may be used on multiple different host platforms, several different versions of the code may be needed for each function. One option is to store these programs in a standard SD memory section of a combo card. Alternately, a standard access means to load the code is contained in the optional Code Storage Area (CSA). The CSA is a separate 16 MB memory area that is accessed using the CSA address pointer and the CSA window register contained in the FBR registers.

24.3.6.1 CSA Access

In order for the host to access a function’s CSA, it first shall determine if that function supports a CSA. The host reads the FBR register at address 0x00n00 where n is the function number (0x1 to 0x7). If bit 6=1, then the function supports a CSA and the host enables access by writing bit 7=1. The next step is for the host to load the 24-bit address to start reading or writing. This is accomplished by writing the 24 bits (A23-0) to registers 0x00n0C to 0x00n0E where n is the function number (0x1 to 0x7). Once the start address is written, data can be read or written by accessing the register 0x00n0F, the CSA data window register. If more than 1 byte needs to be read or written, an extended I/O command (byte or block) can be performed with an OP code of 0 (fixed address). The address pointer shall be automatically incremented with each access to the window register, so the access will be to sequential addresses within the CSA. Once the operation is completed, the address of the next operation shall be held in the 24-bit address register for the host to read.

24.3.6.2 CSA Data Format

The data stored in the CSA shall be structured using the FAT12/FAT16 format. The use of the CSA for program or data storage for different host types requires that the SDIO card manufacturers load the programs and data in a file format that may be recognized by the host. An example of this would be the use of a specific file name saved within a specific subdirectory that is recognized and executed by a particular host operating system. Such formats are specific and sometimes proprietary to different host implementations and operating systems.

24.3.7 SDIO Interrupts

In order to allow the SDIO card to interrupt the host, an interrupt function is added to a pin on the SD interface. Pin number 8, which is used as DAT[1] when operating in the 4-bit SD mode, is used to signal the card’s interrupt to the host. The use of interrupt is optional for each card or function within a card. The SDIO interrupt is “level sensitive”. That is, the interrupt line shall be held active (Low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt cycle. Once the host has serviced the interrupt, it is cleared via some function unique I/O operation. All hosts shall provide pull-up resistors on all data lines DAT[3:0].

24.3.7.1 SPI and SD 1-Bit Mode Interrupts

In the SPI and 1-bit SD mode, Pin 8 is dedicated to the interrupt function. Thus, in the SPI and SD 1-bit modes, there are no timing constraints on interrupts. A card in the SPI or 1-bit SD mode signals an interrupt to the host at any time by asserting pin 8 low. The host detects this pending interrupt using a level sensitive input. The host is responsible for clearing the interrupt. If the SDIO card is operating in the SPI mode, the interrupt from the card may not be asserted if the card is not selected. (CS=0). The exception to this requirement occurs only if the card is both capable of interrupting when not selected (the SCSI bit in the CCCR = 1), and has that feature turned on (the ECSI bit = 1). In this case, the card may assert the interrupt irrespective of the state of the CS line.

24.3.7.2 SD 4-Bit Mode Interrupt

Since Pin 8 is shared between the IRQ and DAT[1] when used in 4-bit SD mode, an interrupt shall only be sent by the card and recognized by the host during a specific time. The time that a low level on Pin 8 shall be recognized as an interrupt is defined as the interrupt cycle. An SDIO host shall only sample the level on Pin 8 (DAT[1]/IRQ) into the interrupt detector during the Interrupt Period. At all other times, the host interrupt controller shall ignore the level on Pin 8. Note that the interrupt cycle is applicable for both memory and I/O operations. The definition of the interrupt cycle is different for operations with single block and multiple block data transfer.

24.3.7.3 Interrupt Clear Timing

Since the SDIO card uses level sensitive interrupts, the host shall clear pending interrupts with an I/O read or write to some function unique area. In some host implementations, the sending of a CMD52 to the card is handled by host adapter hardware while the host CPU can execute other operations. This condition may allow an interrupt that has already been handled to re-interrupt the host if the timing of the interrupt clear is not controlled. To prevent this condition, any SDIO card that implements interrupts shall follow some required timing with respect to removing the interrupt from the DAT[1] line after the write to the function unique area that clears the interrupt. The clearing of the interrupt can be caused by an I/O write in a function unique method, or by a function unique I/O read. An example of clearing an interrupt using an I/O read would be a function where the reading of a data register may automatically clear the data ready interrupt.

24.3.8 SDIO Suspend/Resume Operation

The procedure used to perform the Suspend/Resume operation on the SD bus is:

- The host determines which function is currently using the DAT[3:0] line(s).
- The host requests the lower priority or slower transaction to suspend.
- The host checks for the transaction suspension to complete.
- The host begins the higher priority transaction.

- The host waits for the completion of the higher priority transaction.
- The host restores the suspended transaction.

If the current transaction can accept suspend and the card receives a Suspend command during Read Wait, it shall accept the Suspend request.

24.3.9 SDIO Read Wait Operation

The optional Read Wait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The read Wait operation allows a host to signal a card that is executing a read multiple (CMD53) operation to temporarily stall the data transfer while allowing the host to send commands to any function within the SDIO card. To determine if a card supports the Read Wait protocol, the host shall test the SRW capability bit in the card capability byte of the CCCR. The timing for Read Wait is based on the interrupt cycle. If a card does not support the Read Wait protocol, the only means a host has to stall (not abort) data in the middle of a read multiple command is to control the SDCLK. Read Wait support is mandatory for the card to support Suspend/Resume.

25.1 Overview

Low power consumption is an important indicator of IoT applications. The chip's CPU supports the working mode, idle power saving mode, and sleep mode. You can select a proper mode according to the current application scenario to reduce the chip's power consumption and prolong the battery life.

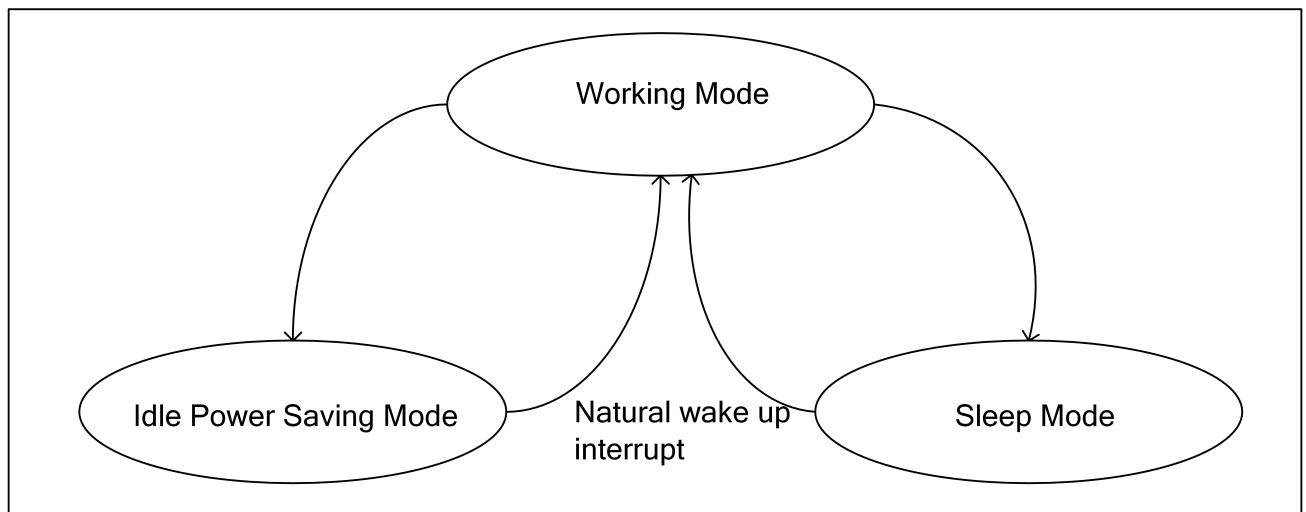


Fig. 25.1: Low power modes

25.2 Features

- Clock control: clock control of peripherals in GLB, small-scale power saving, and fast response speed
- Sleep control (PDS): 5 levels (PDS 1/2/3/7/15), large-scale power saving, moderate response speed
- Deep sleep control (HBN): 4 levels (HBN 0/1/2/3), global power saving, and long response time

25.3 Functional Description

25.3.1 Power Domain

There are 8 power domains in the xxx chip, with main functions as follows:

- PD_AON
 - HBN state machine, which can control the power/isolation unit/reset/clock of PD_AON_HBNRTC/PD_AON_-HBNCORE/PD_CORE
 - AON_PIN (GPIO16/17/18/19) pin wakeup function
 - Reserve 4 readable and writable registers to save data in PDS/HBN0/HBN1/HBN2 modes
 - BOR (Brown Out Reset) function
 - LDO11_AON/RT/SOC output voltage selection unit
 - Root, F32K, and Uart clock source selection
 - HBN_OUT0_PIR (Acomp0/Acomp1/bor/pir) wakeup mask, enable register, and interrupt status register
- PD_AON_HBNRTC
 - Select the control unit of RTC Counter clock source
 - RTC can be used for wakeup or LED flashing
 - RC32K and Xtal32X control functions
- PD_AON_HBNCORE
 - Partial power control register
 - Reserve HBN_RAM, which can be used to store programs and data, so that data will not disappear in PDS/HBN0 modes
 - In HBN mode, it has the function of controlling AON_PIN and keeping other IO
 - PIR digital control: PIR is a Passive Infra-Red (PIR) sensor, a peripheral in HBN area, which can be used as HBN wakeup source
 - Reserve LDO11SOC, LDO15RF, DCDC0, DCDC1 and DCDC2 control registers

- Reserve XTAL, TSEN, Acomp0/1 and GPADC control registers
- Reserve 4 readable and writable registers to save data in PDS/HBN0 modes
- PD_CORE
 - The PDS state machine controls the power, isolation unit, reset, clock, and memory of PD_CORE_MISC/PD_USB/PD_CPU/PD_WB
 - Reserve PDS_RAM, so that data will not disappear in PDS mode
 - WIFI/BLE timer control
 - 160KB WRAM Retention/Sleep
 - Control the PDS interrupt and wakeup function
 - Reserve the control of GPIO0~15/20~36
 - Reserve the control of RC32M
 - PDS_Timer timer
- PD_CORE_MISC
 - PD_CORE_MISC_DIG and PD_CORE_MISC_ANA are collectively referred to PD_CORE_MISC
 - Peripheral (including I2C/SDIO/UART/FLASH/SPI/ADC/DAC/GPIO/PWM)
 - Chip GLB register
- PD_USB
 - USB controller
- PD_CPU
 - NP_CPU and cache unit
 - ROM, TCM
- PD_WB
 - WIFI PHY/MAC
 - BLE PHY/MAC
 - RF Controller

Each power domain is controlled by 8 different power modes as shown below:

Table 25.1: Power mode

NO.	Scenario	Power Domain							
		PD_AON	PD_AON_-HBNRTC	PD_-AON_HB-NCORE	PD_-CORE	PD_CORE_-MISC	PD_USB	PD_CPU	PD_WB
1	Normal	ON	ON	ON	ON	ON	ON	ON	ON
2	PDS1	ON	ON	ON	ON	ON	ON	ON	OFF
3	PDS2	ON	ON	ON	ON	ON	ON	OFF	ON
4	PDS3	ON	ON	ON	ON	ON	ON	OFF	OFF
5	PDS7	ON	ON	ON	ON	ON	OFF	OFF	OFF
6	PDS15	ON	ON	ON	ON	OFF	OFF	OFF	OFF
7	HBN0	ON	ON	ON	OFF	OFF	OFF	OFF	OFF
8	HBN1	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF
9	HBN2	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
10	HBN3	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

25.3.2 Wake-up Source

The chip supports multiple wakeup sources, which can wake up the chip from different power modes.

The wake-up sources for different power modes are shown in the following table:

Table 25.2: Wakeup source

Power mode	Wakeup source
PDS0	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer/GPIO/IRRX/DM/USB/WIFI
PDS1	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer/GPIO/IRRX/DM/USB
PDS2	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer/GPIO/IRRX/DM/USB/WIFI
PDS3	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer/GPIO/IRRX/DM/USB
PDS7	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer/GPIO/IRRX/DM
PDS15	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer
HBN0	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1
HBN1	AON_PIN/RTC
HBN2	AON_PIN
HBN3	Reapply power to VDDIO2

25.3.3 Power Modes

Operating mode

The chip provides independent clock control between CPU and peripherals. The GLB and clock sections detail the clock control of each module. The software can perform clock control over CPUs or peripherals that are not in use according to the current application scenario. The clock control provides real-time response, so there is no need to worry about the response time in this working mode.

Power-down sleep mode

The power-down mode consumes less power than the working mode. In the PDS mode, the clocks other than RTC will be controlled and switched to the internal low-speed clocks, and the external crystal oscillator and PLL will be turned off to save more power, so there will be a time delay when entering and exiting this low-power mode. After the power-down sleep mode is enabled, the data in the OCRAM area can automatically switch to the “retention” state and remain, and automatically exit this state after wake-up.

1. Enter Idle Power Saving Mode

The software can make this module enter the power-down mode through PDS configuration, waiting for processing. After entering the wait for interrupt (WFI) mode, PDS will trigger the clock control module to perform the gate clock operation, and notify the analog circuit to turn off the PLL and external crystal oscillator

2. Exit Idle Power Saving Mode

There are two ways to exit the idle power saving mode. One is that a specific interrupt or event stops the idle state. The other is that the time in PDS_TIM set by the software is met. Both will trigger PDS to exit the power-down mode. Considering that it takes about 1 ms to turn on the crystal oscillator, PDS allows software to turn on the crystal oscillator in advance, to wake up PDS faster. When PDS is ready to wake up, it will notify CPU to exit the WFI mode through interrupt.

HBN

In the sleep mode, most of the chip logic is powered off (Vcore) while the AON power is kept ON, and the internal circuit will not wake up until an external event is received. This mode can achieve ultimate power saving, but it takes the longest response time compared with the first two modes, so it suits the case where the chip does not need to work for a long time, to prolong the battery life. As most circuits will be powered off in this mode, the corresponding register values and memory data will disappear. Therefore, there is a 4 KB HBN_RAM reserved in HBN that will not be powered off in sleep state. The data or state that the software needs to save can be copied to this memory before the chip enters the sleep mode. When recovering from the sleep mode, the chip can access data directly from RAM, which can usually be used as a record of state or a quick data recovery.

25.3.4 IO Retention

IO retention includes AON_IO retention and PDS_IO retention. In the PDS 1/2/3/7 mode, for the chip's MISC domain is still powered, GPIO can be controlled by the glb register. After the glb register is powered off, AON_CTRL and PDS can control the IE/PD/PU of AON_IO and PDS_IO.

AON_IO

AON_IO refers to GPIO16/17/18/19. GPIO16/17 can be used as XTAL32K input and output.

When reg_en_aon_ctrl_gpio is 1, reg_en_aon_ctrl_gpio[3:0] controls whether GPIO16/17/18/19 are controlled by AON_HW. When reg_en_aon_ctrl_gpio is 1, the pull-up enable of AON_IO is controlled by reg_aon_gpio_pu, the pull-down enable is controlled by reg_aon_gpio_pd, IE/SMT is controlled by reg_aon_gpio_ie_smt, and OE is controlled by reg_aon_gpio_oe.

1. Hardware IO retention HBN can control the IE/PD/PU/OE/O of AON_IO to achieve IO retention. When reg_en_aon_ctrl_gpio is 1, the pull-up enable of AON_IO is controlled by reg_aon_pad_pu, the pull-down enable is controlled by reg_aon_pad_pd, OE is controlled by reg_aon_pad_oe, and IE/SMT is controlled by reg_aon_pad_ie_smt. For example, when reg_en_aon_ctrl_gpio is 0 and reg_aon_pad_pu is 1, the pull-up function cannot be implemented; when reg_aon_gpio_ie_smt is 1 and reg_aon_pad_pu is 1, the pull-up function can be implemented.

2. Software IO keep After setting reg_aon_gpio_iso_mode to 1, when entering HBN mode, AON PAD can keep OEO, but PUPD cannot keep it; after HBN wakes up, AON PAD state will still keep, you need to set reg_aon_gpio_iso_mode to 0 before leaving IO On hold. For example, GPIO16 keeps a high level in HBN mode, it needs to be configured as a normal IO function first, then use the glb register to configure it to output a high level, and finally configure reg_aon_gpio_iso_mode to 1, then enter the HBN mode.

PDS_IO

PDS_IO refers to other GPIOs except AON_IO, a total of 30 GPIOs, divided into 2 groups:

- GPIO0~15
- GPIO20~33

Note that GPIO16~19 can be controlled by PDS to achieve software IO retention, but cannot achieve hardware IO retention.

1. Hardware IO retention The IE/PD/PU of PDS_IO can be controlled by the pds_gpio_i_set register, the same group of GPIOs must maintain the same level. For example, if GPIO0 is configured as a pull-up, then GPIO8 is also configured as a pull-up.

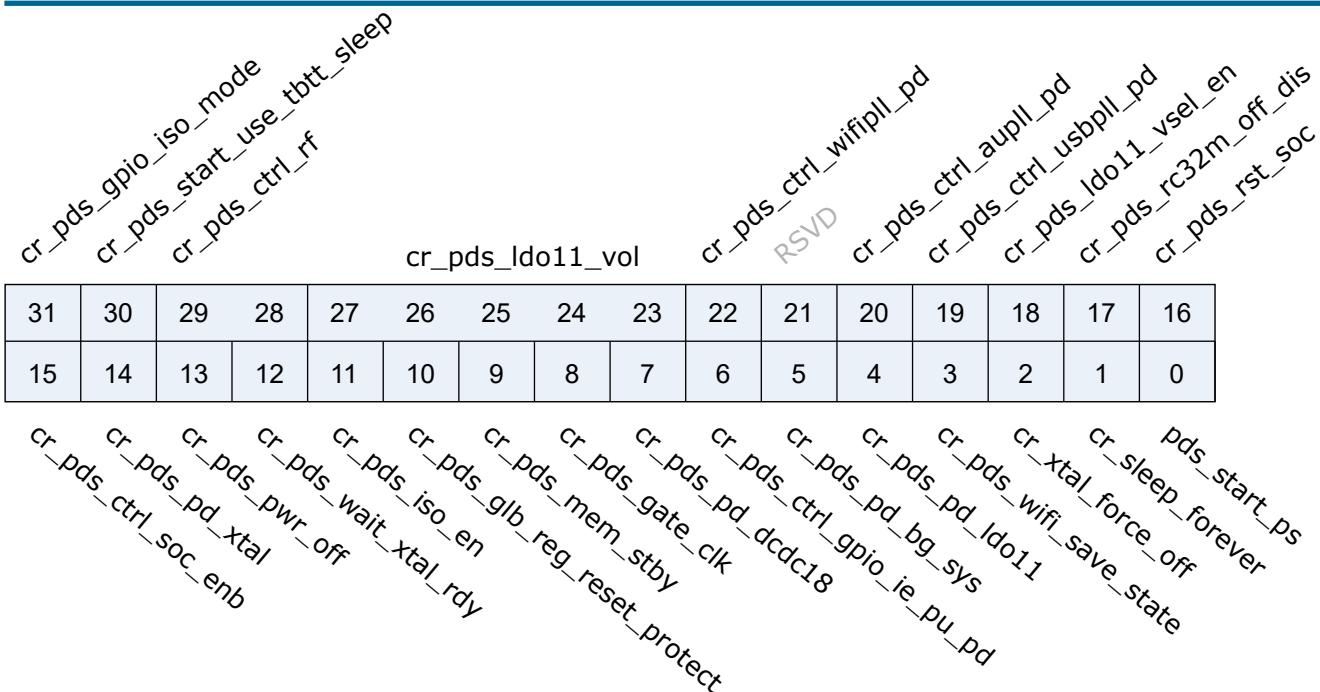
2. Software IO keep When cr_pds_gpio_iso_mode is 1, after entering PDS7 mode, if cr_pds_gpio_kee_en[0], [1], [2] are 1, GPIO0~15, GPIO20~33 (excluding GPIO21/22/28/29), GPIO16~19 respectively enter the GPIO hold state. After the PDS wakes up, the PDS_IO state will still be maintained. You need to set cr_pds_gpio_iso_mode to 0 before leaving the IO hold state. The advantage of this IO retention method is that the same group of GPIOs can be kept at different levels.

25.4 Register description

Name	Description
PDS_CTL	
PDS_TIME1	
PDS_INT	
PDS_CTL2	
PDS_CTL3	
PDS_CTL4	
pds_stat	
pds_ram1	
PDS_CTL5	
PDS_RAM2	
pds_gpio_i_set	
pds_gpio_pd_set	
pds_gpio_int	
pds_gpio_stat	
PDS_RAM3	
PDS_RAM4	

25.4.1 PDS_CTL

Address: 0x2000e000



Bits	Name	Type	Reset	Description
31	<code>cr_pds_gpio_iso_mode</code>	r/w	0	1: HW Keep GPIO @ PDS7 0 : Clear this bit after PDS7 to release GPIO
30	<code>RSVD</code>			
29:28	<code>cr_pds_ctrl_rf</code>	r/w	2'b01	00 : PDS don't control RF on/off 01 : PDS control RF on/off depend on misc_pwr_off 10 : PDS control RF on/off depend on any power off 11 : PDS control RF on/off whe pds at idle state
27:23	<code>cr_pds_ldo11_vol</code>	r/w	5'h8	LDO11 voltage value in PDS mode output voltage sel: 0: 0.9V, 4: 1.0V, 8: 1.1V, 12: 1.2V, 25mV/step
22	<code>cr_pds_ctrl_wifipl_pd</code>	r/w	0	PDS Control WIFI PLL off When pds_pwr_off
21	<code>RSVD</code>			
20	<code>cr_pds_ctrl_aupll_pd</code>	r/w	0	PDS Control Audio PLL off When pds_pwr_off
19	<code>cr_pds_ctrl_usbpll_pd</code>	r/w	0	PDS Control USB PLL off When pds_pwr_off
18	<code>cr_pds_ldo11_vsel_en</code>	r/w	0	PDS "SLEEP" control LDO11 voltage enable
17	<code>cr_pds_rc32m_off_dis</code>	r/w	0	1 : RC32M always on @any state 0 : RC32M on/off controlled by PDS state
16	<code>cr_pds_rst_soc</code>	r/w	0	0 : no pds_reset 1: pds_RST controlled by PDS
15	<code>cr_pds_ctrl_soc_enb</code>	r/w	0	1 : pds_soc_enb controlled by PDS 0 : pds_soc_enb always active

Bits	Name	Type	Reset	Description
14	cr_pds_pd_xtal	r/w	1	0 : don't_touch xtal during PDS 1 : xtal power down during PDS
13	cr_pds_pwr_off	r/w	1	0 : don't_touch Power during PDS 1 : Power off during PDS (each power domain can has its own control)
12	cr_pds_wait_xtal_rdy	r/w	0	0 : Skip wait XTAL Ready before PDS Interrupt 1 : wait XTAL Ready during before PDS Interrupt
11	cr_pds_iso_en	r/w	1	0 : don't_touch Isolation during PDS (all power domain) 1 : Isolation during PDS (each power domain can has its own control)
10	cr_pds_glb_reg_reset_protect	r/w	0	1: avoid glb_reg reset by any reset
9	cr_pds_mem_stby	r/w	1	0 : don't_touch mem_stby during PDS 1 : mem_stby during PDS (each power domain can has its own control)
8	cr_pds_gate_clk	r/w	1	0 : don't_touch clock gating during PDS (all power domain) 1 : gate clock during PDS (each pwr domain has its own control)
7	cr_pds_pd_dcdc18	r/w	0	0 : don't_touch dcdc18 during PDS 1 : power down dcdc18 during PDS
6	cr_pds_ctrl_gpio_ie_pu_pd	r/w	0	1: allow PDS Control the GPIO IE/PU/PD at Sleep Mode
5	cr_pds_pd_bg_sys	r/w	0	0 : don't_touch bg_sys during PDS 1 : power down bg_sys during PDS
4	cr_pds_pd_ldo11	r/w	0	0 : don't_touch ldo11 during PDS 1 : power down ldo11 during PDS
3	cr_pds_wifi_save_state	r/w	0	Save WIFI State Before Enter PDS
2	cr_xtal_force_off	r/w	0	
1	cr_sleep_forever	r/w	0	
0	pds_start_ps	w1p	0	Enter PDS

25.4.2 PDS_TIME1

Address: 0x2000e004

cr_sleep_duration

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

cr_sleep_duration

Bits	Name	Type	Reset	Description
31:0	cr_sleep_duration	r/w	32'd3240	PDS Sleep Time (in units of 32K clock cycles)

25.4.3 PDS_INT

Address: 0x2000e00c

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

ro_pds_wakeup_event

RSVD

RSVD

cr_pds_wakeup_src_en

cr_pds_wakeup_src_en

RSVD

cr_pds_int_clr

cr_pds_rf_done_int_mask

cr_pds_wifi_tbtt_sleep_irq_mask

cr_pds_wifi_tbtt_wakeup_irq_mask

ro_pds_rf_done_int

ro_pds_wifi_tbtt_sleep_irq

ro_pds_rf_done_int

ro_pds_wifi_tbtt_wakeup_irq

ro_pds_rf_done_int

ro_pds_rf_done_int

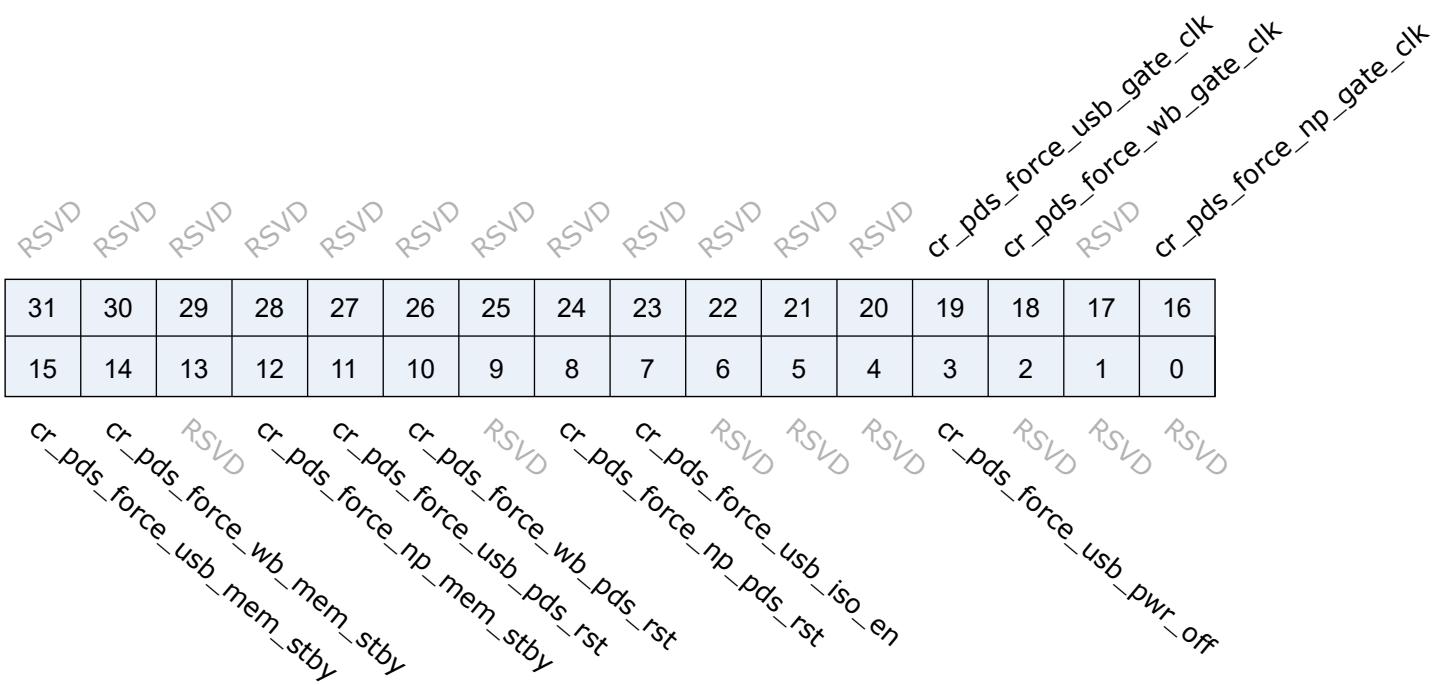
ro_pds_rf_done_int

Bits	Name	Type	Reset	Description
31:9	RSVD			
8	cr_pds_int_clr	r/w	0	pds interrupt clear
7:6	RSVD			
5	cr_pds_rf_done_int_mask	r/w	0	Mask pds rf done interrupt
4	cr_pds_wake_int_mask	r/w	0	Mask pds wakeup interrupt

Bits	Name	Type	Reset	Description
3:2	RSVD			
1	ro_pds_rf_done_int	r	0	pu_rf_done interrupt
0	ro_pds_wake_int	r	0	PDS Wakeup Interrupt

25.4.4 PDS_CTL2

Address: 0x2000e010

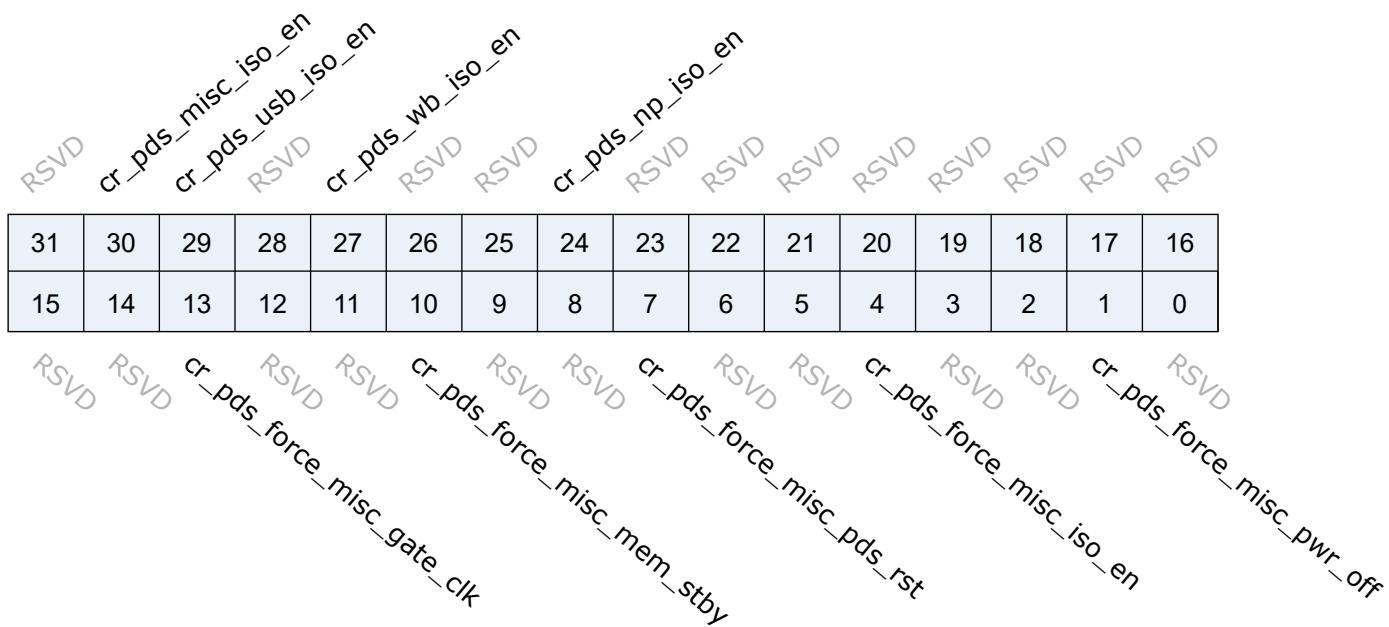


Bits	Name	Type	Reset	Description
31:20	RSVD			
19	cr_pds_force_usb_gate_clk	r/w	0	manual force usbio clock gated
18	cr_pds_force_wb_gate_clk	r/w	0	manual force WB clock gated
17	RSVD			
16	cr_pds_force_np_gate_clk	r/w	0	manual force NP clock gated
15	cr_pds_force_usb_mem_stby	r/w	0	manual force usbio memory sleep
14	cr_pds_force_wb_mem_stby	r/w	0	manual force WB memory sleep
13	RSVD			
12	cr_pds_force_np_mem_stby	r/w	0	manual force NP memory sleep
11	cr_pds_force_usb_pds_rst	r/w	0	manual force usbio pds reset

Bits	Name	Type	Reset	Description
10	cr_pds_force_wb_pds_RST	r/w	0	manual force WB pds reset
9	RSVD			
8	cr_pds_force_np_pds_RST	r/w	0	manual force NP pds reset
7	cr_pds_force_usb_iso_en	r/w	0	manual force usbio isolation
6	cr_pds_force_wb_iso_en	r/w	0	manual force WB isolation
5	RSVD			
4	cr_pds_force_np_iso_en	r/w	0	manual force NP isolation
3	cr_pds_force_usb_pwr_off	r/w	0	manual force usbio power off
2	cr_pds_force_wb_pwr_off	r/w	0	manual force WB power off
1	RSVD			
0	cr_pds_force_np_pwr_off	r/w	0	manual force NP power off

25.4.5 PDS_CTL3

Address: 0x2000e014



Bits	Name	Type	Reset	Description
31	RSVD			
30	cr_pds_misc_iso_en	r/w	1	1 : make misc isolated at PDS Sleep state 0 : make misc isolated at PDS Sleep state

Bits	Name	Type	Reset	Description
29	cr_pds_usb_iso_en	r/w	1	1 : make usb isolated at PDS Sleep state 0 : make usb isolated at PDS Sleep state
28	RSVD			
27	cr_pds_wb_iso_en	r/w	1	1 : make WB isolated at PDS Sleep state 0 : make WB isolated at PDS Sleep state
26:25	RSVD			
24	cr_pds_np_iso_en	r/w	1	1 : make NP isolated at PDS Sleep state 0 : make NP isolated at PDS Sleep state
23:14	RSVD			
13	cr_pds_force_misc_gate_clk	r/w	0	manual force MISC gate_clk
12:11	RSVD			
10	cr_pds_force_misc_mem_stby	r/w	0	manual force MISC mem_stby
9:8	RSVD			
7	cr_pds_force_misc_pds_RST	r/w	0	manual force MISC pds_RST
6:5	RSVD			
4	cr_pds_force_misc_iso_en	r/w	0	manual force MISC iso_en
3:2	RSVD			
1	cr_pds_force_misc_pwr_off	r/w	0	manual force MISC pwr_off
0	RSVD			

25.4.6 PDS_CTL4

Address: 0x2000e018

RSVD	RSVD	RSVD	RSVD	cr_pds_misc_gate_clk	cr_pds_misc_mem_stby	cr_pds_misc_reset	cr_pds_misc_pwr_off	cr_pds_usb_gate_clk	cr_pds_usb_mem_stby	cr_pds_usb_reset	cr_pds_usb_pwr_off	RSVD	RSVD	RSVD	RSVD	RSVD	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		

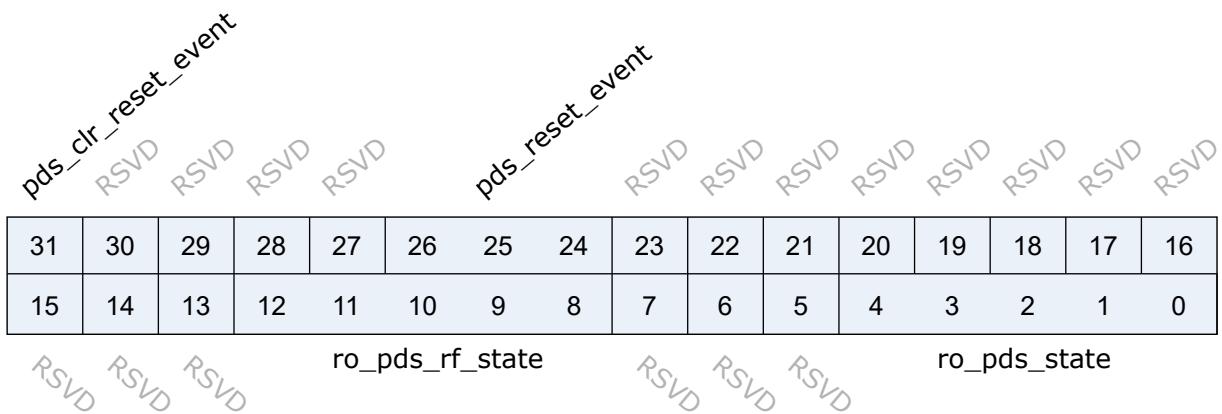
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
cr_pds_wb_gate_clk	cr_pds_wb_mem_stby	RSVD	cr_pds_np_pwr_off	cr_pds_np_mem_stby	cr_pds_np_gate_clk	cr_pds_np_reset	cr_pds_np_gate_clk	cr_pds_wb_mem_stby									

Bits	Name	Type	Reset	Description
31:28	RSVD			
27	cr_pds_misc_gate_clk	r/w	1	1 : make core_misc clock gated at PDS Sleep state 0 : make core_misc clocking at PDS Sleep state
26	cr_pds_misc_mem_stby	r/w	1	1 : make core_misc RAM @Retention at PDS Sleep state 0 : make core_misc RAM @ Normal at PDS Sleep state
25	cr_pds_misc_reset	r/w	1	1 : make core_misc reset at PDS Sleep state 0 : make core_misc not reset at PDS Sleep state
24	cr_pds_misc_pwr_off	r/w	1	1 : make core_misc Power off at PDS Sleep state 0 : make core_misc power on at PDS Sleep state
23	cr_pds_usb_gate_clk	r/w	1	1 : make usb clock gated at PDS Sleep state 0 : make usb clocking at PDS Sleep state
22	cr_pds_usb_mem_stby	r/w	1	1 : make usb RAM @Retention at PDS Sleep state 0 : make usb RAM @ Normal at PDS Sleep state
21	cr_pds_usb_reset	r/w	1	1 : make usb reset at PDS Sleep state 0 : make usb not reset at PDS Sleep state
20	cr_pds_usb_pwr_off	r/w	1	1 : make usb Power off at PDS Sleep state 0 : make usb power on at PDS Sleep state
19:16	RSVD			
15	cr_pds_wb_gate_clk	r/w	1	1 : make WB clock gated at PDS Sleep state 0 : make WB clocking at PDS Sleep state
14	cr_pds_wb_mem_stby	r/w	1	1 : make WB RAM @Retention at PDS Sleep state 0 : make WB RAM @ Normal at PDS Sleep state

Bits	Name	Type	Reset	Description
13	cr_pds_wb_reset	r/w	1	1 : make WB reset at PDS Sleep state 0 : make WB not reset at PDS Sleep state
12	cr_pds_wb_pwr_off	r/w	1	1 : make WB Power off at PDS Sleep state 0 : make WB power on at PDS Sleep state
11:4	RSVD			
3	cr_pds_np_gate_clk	r/w	1	1 : make NP clock gated at PDS Sleep state 0 : make NP clocking at PDS Sleep state
2	cr_pds_np_mem_stby	r/w	1	1 : make NP RAM @Retention at PDS Sleep state 0 : make NP RAM @ Normal at PDS Sleep state
1	cr_pds_np_reset	r/w	1	1 : make NP reset at PDS Sleep state 0 : make NP not reset at PDS Sleep state
0	cr_pds_np_pwr_off	r/w	1	1 : make NP Power off at PDS Sleep state 0 : make NP power on at PDS Sleep state

25.4.7 pds_stat

Address: 0x2000e01c

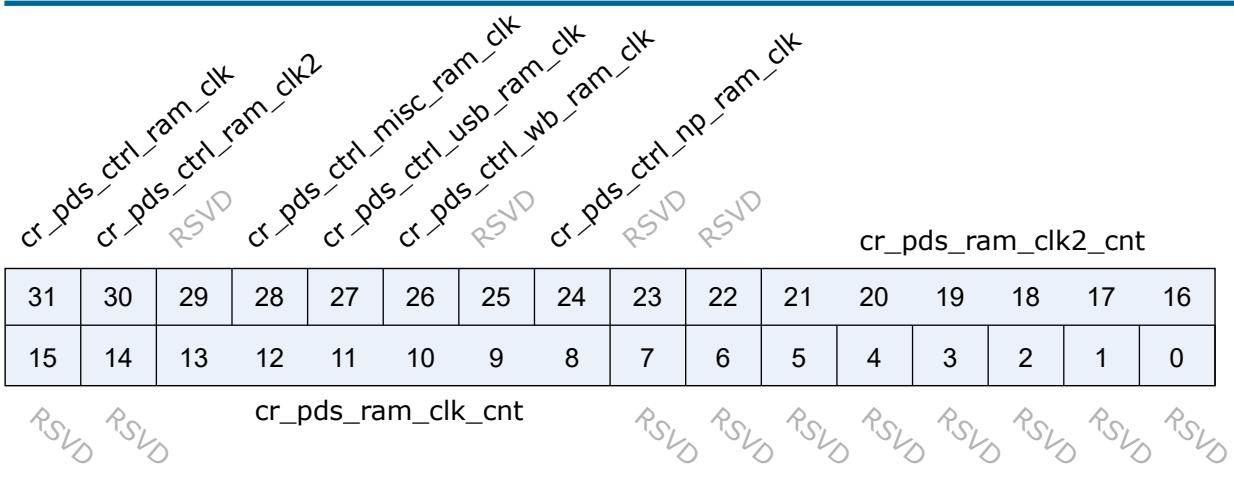


Bits	Name	Type	Reset	Description
31	pds_clr_reset_event	w1c	0	clear pds reset event
30:27	RSVD			
26:24	pds_reset_event	r	0	[2] : pds_rst_n (pds reset) [1]: pwr_rst_n (hbn power on reset) [0]: hreset_n (Bus Reset)
23:13	RSVD			

Bits	Name	Type	Reset	Description
12:8	ro_pds_rf_state	r	5'b0	ST_PDS_RF_OFF = 5'b0000 ; ST_PDS_PU_MBG = 5'b0001 ; ST_PDS_PU_LDO15RF = 5'b0011 ; ST_PDS_PU_SFREG = 5'b0111 ; ST_PDS_PUD_XTAL18 = 5'b01111 ; ST_PDS_WB_EN_AON = 5'b11111 ;
7:5	RSVD			
4:0	ro_pds_state	r	5'b0	ST_IDLE = 5'b00000; ST_MEM_STBY = 5'b10000; ST_ECG = 5'b01000; ST_ERST = 5'b01100; ST_EISO = 5'b01111; ST_POFF = 5'b00111; ST_PRE_BGON = 5'b00011; ST_PRE_BGON1 = 5'b00001; ST_BGON = 5'b00101; ST_CLK_SW_32M= 5'b00100; ST_PON_DCDC = 5'b00110; ST_PON_LDO11_MISC = 5'b01110; ST_PON = 5'b01010; ST_DISO = 5'b00010; ST_DCG = 5'b01101; ST_MEM_IDLE = 5'b11000; ST_DRST = 5'b01011; ST_WAIT_EFUSE= 5'b01001;

25.4.8 pds_ram1

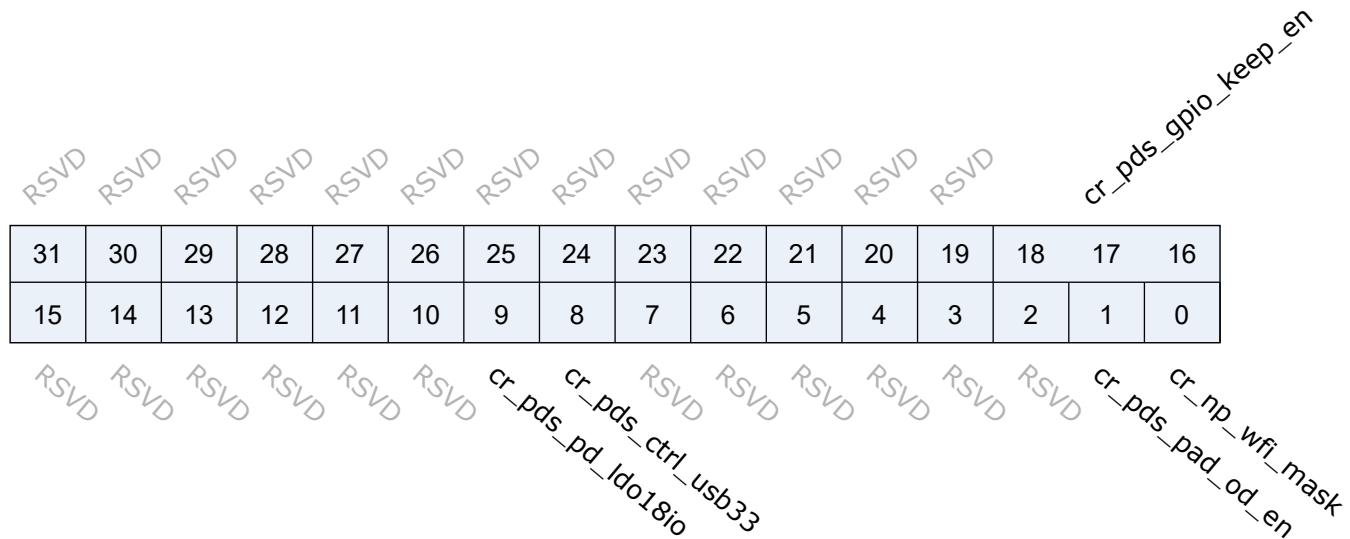
Address: 0x2000e020



Bits	Name	Type	Reset	Description
31	cr_pds_ctrl_ram_clk	r/w	1'b0	1 : Enable PDS Control PD_CORE SRAM Clock @ PDS Sequence
30	cr_pds_ctrl_ram_clk2	r/w	1'b0	HW Option To assert extra clock during PDS on sequence
29	RSVD			
28	cr_pds_ctrl_misc_ram_clk	r/w	1'b0	This bit is Enable by bit [31] : cr_pds_ctrl_ram_clk 1 : PDS Control PD_CORE_MISC SRAM Clock @ PDS Sequence 0 : PDS do nothing on SRAM Clock
27	cr_pds_ctrl_usb_ram_clk	r/w	1'b0	This bit is Enable by bit [31] : cr_pds_ctrl_ram_clk 1 : PDS Control PD_usb SRAM Clock @ PDS Sequence 0 : PDS do nothing on PD_usb SRAM Clock
26	cr_pds_ctrl_wb_ram_clk	r/w	1'b0	This bit is Enable by bit [31] : cr_pds_ctrl_ram_clk 1 : PDS Control PD_WB SRAM Clock @ PDS Sequence 0 : PDS do nothing on PD_WB SRAM Clock
25	RSVD			
24	cr_pds_ctrl_np_ram_clk	r/w	1'b0	This bit is Enable by bit [31] : cr_pds_ctrl_ram_clk 1 : PDS Control PD_CORE_CPU SRAM Clock @ PDS Sequence 0 : PDS do nothing on PD_CORE_CPU SRAM Clock
23:22	RSVD			
21:16	cr_pds_ram_clk2_cnt	r/w	6'd24	HW Option : Assert Extra Clock Counter in MEM_IDLE
15:14	RSVD			
13:8	cr_pds_ram_clk_cnt	r/w	6'd8	HW Option : Assert Extra Clock Counter in MEM_STBY
7:0	RSVD			

25.4.9 PDS_CTL5

Address: 0x2000e024



Bits	Name	Type	Reset	Description
31:19	RSVD			
18:16	cr_pds_gpio_keep_en	r/w	3'b111	if cr_pds_gpio_iso_mode=1, can use bit to enable or disable keep function [0] : GPIO0 15 [1] : GPIO20 36 (not include GPIO21/22/28/29) [2] : GPIO16 19
15:10	RSVD			
9	cr_pds_pd_ldo18io	r/w	0	0 : don't_touch ldo18io during PDS 1 : power down ldo18io during PDS
8	cr_pds_ctrl_usb33	r/w	0	Set this bit to enable HW control turn on/off USB 3.3V @USB1.1V Power On/OFF (Replace the function of reg_pu_usb20_psw)
7:2	RSVD			
1	cr_pds_pad_od_en	r/w	0	GPIO21/22/28/29 5V Tolerant PAD Open Drain Enable
0	cr_np_wfi_mask	r/w	0	pds start condition mask np_wfi

25.4.10 PDS_RAM2

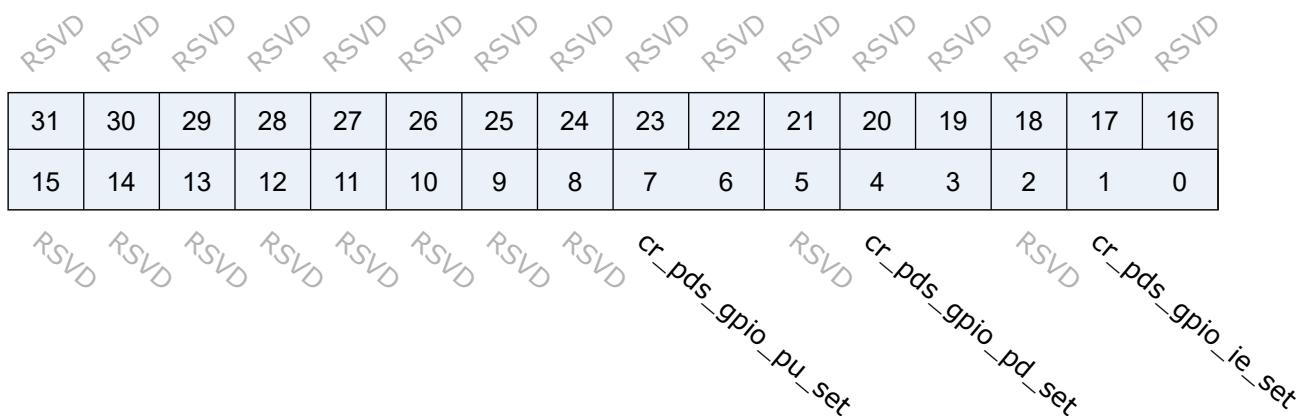
Address: 0x2000e028

RSVD	cr_wram_ret																
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		

Bits	Name	Type	Reset	Description
31:20	RSVD			
19:10	cr_wram_ret	r/w	10'h0	[9] : 144 160KB WRAM Retention [8] : 128 144KB WRAM Retention [7] : 112 128KB WRAM Retention [6] : 96 112KB WRAM Retention [5] : 80 96KB WRAM Retention [4] : 64 80KB WRAM Retention [3] : 48 64KB WRAM Retention [2] : 32 48KB WRAM Retention [1] : 16 32KB WRAM Retention [0] : 0 16KB WRAM Retention
9:0	cr_wram_slp	r/w	10'h0	[9] : 144 160KB WRAM SLEEP [8] : 128 144KB WRAM SLEEP [7] : 112 128KB WRAM SLEEP [6] : 96 112KB WRAM SLEEP [5] : 80 96KB WRAM SLEEP [4] : 64 80KB WRAM SLEEP [3] : 48 64KB WRAM SLEEP [2] : 32 48KB WRAM SLEEP [1] : 16 32KB WRAM SLEEP [0] : 0 16KB WRAM SLEEP

25.4.11 pds_gpio_i_set

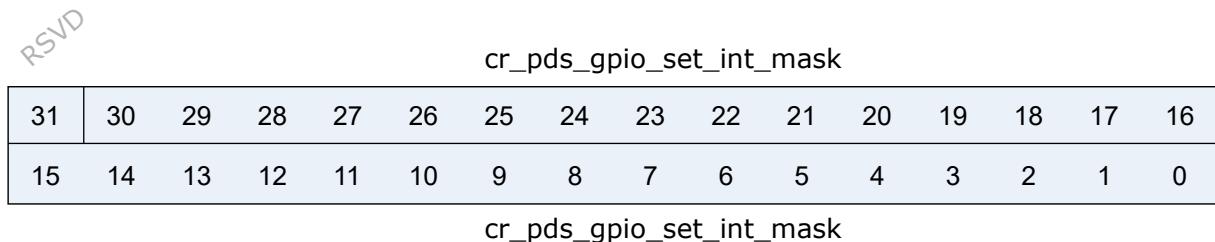
Address: 0x2000e030



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:6	cr_pds_gpio_pu_set	r/w	2'b0	Enable GPIO PU @ PDS [0] : GPIO0 15 [1] : GPIO20 36
5	RSVD			
4:3	cr_pds_gpio_pd_set	r/w	2'b0	Enable GPIO PD @ PDS [0] : GPIO0 15 [1] : GPIO20 36
2	RSVD			
1:0	cr_pds_gpio_ie_set	r/w	2'b0	Enable GPIO IE @ PDS [0] : GPIO0 15 [1] : GPIO20 36

25.4.12 pds_gpio_pd_set

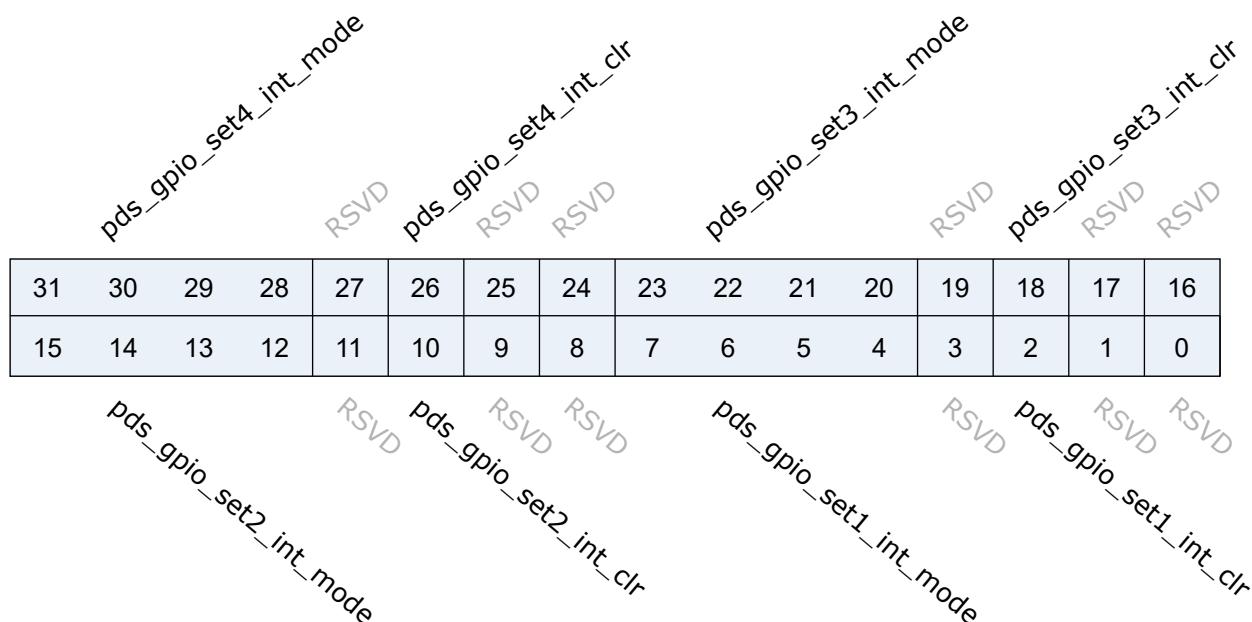
Address: 0x2000e034



Bits	Name	Type	Reset	Description
31	RSVD			
30:0	cr_pds_gpio_set_int_mask	r/w	31'h7FFFFFFF	PDS Interrupt Mask for GPIO <ul style="list-style-type: none"> [0] GPIO0 [1] GPIO1 [2] GPIO2 [3] GPIO3 [4] GPIO4 [5] GPIO5 [6] GPIO6 [7] GPIO7 [8] GPIO8 [9] GPIO9 [10] GPIO10 [11] GPIO11 [12] GPIO12 [13] GPIO13 [14] GPIO14 [15] GPIO15 [16] GPIO20 [17] GPIO21 [18] GPIO22 [19] GPIO23 [20] GPIO24 [21] GPIO25 [22] GPIO26 [23] GPIO27 [24] GPIO28 [25] GPIO29 [26] GPIO30 [27] GPIO31 [28] GPIO32 [29] GPIO33 [30] GPIO34

25.4.13 pds_gpio_int

Address: 0x2000e040



Bits	Name	Type	Reset	Description
31:28	pds_gpio_set4_int_mode	r/w	4'b0	GPIO28 34 PDS Interrupt Mode 0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
27	RSVD			
26	pds_gpio_set4_int_clr	r/w	1'b0	Clear GPIO28 34 PDS IO Interrupt
25:24	RSVD			

Bits	Name	Type	Reset	Description
23:20	pds_gpio_set3_int_mode	r/w	4'b0	GPIO20 27 PDS Interrupt Mode 0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
19	RSVD			
18	pds_gpio_set3_int_clr	r/w	1'b0	Clear GPIO20 27 PDS IO Interrupt
17:16	RSVD			
15:12	pds_gpio_set2_int_mode	r/w	4'b0	GPIO8 15 PDS Interrupt Mode 0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
11	RSVD			
10	pds_gpio_set2_int_clr	r/w	1'b0	Clear GPIO8 15 PDS IO Interrupt
9:8	RSVD			
7:4	pds_gpio_set1_int_mode	r/w	4'b0	GPIO0 7 PDS Interrupt Mode 0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
3	RSVD			
2	pds_gpio_set1_int_clr	r/w	1'b0	Clear GPIO0 7 PDS IO Interrupt

Bits	Name	Type	Reset	Description
1:0	RSVD			

25.4.14 pds_gpio_stat

Address: 0x2000e044

pds_gpio_int_stat																
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	
pds_gpio_int_stat																

Bits	Name		Type	Reset	Description											
31	RSVD															
30:0	pds_gpio_int_stat		r	31'b0												

25.4.15 PDS_RAM3

Address: 0x2000e048

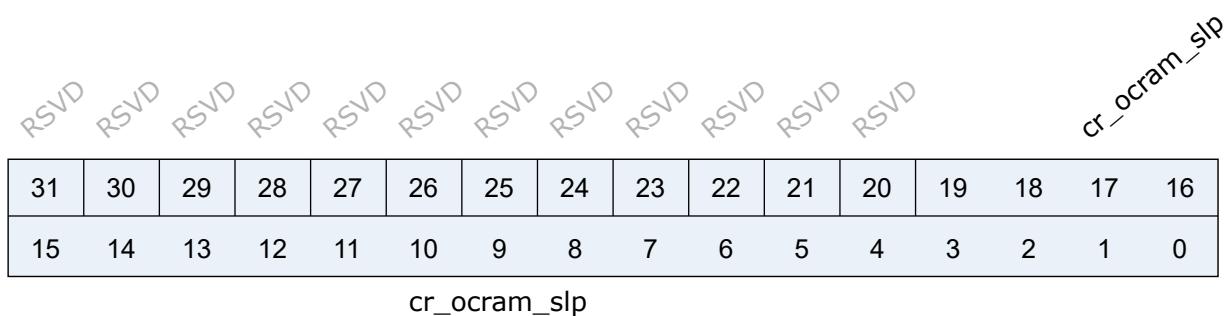
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
cr_ocram_ret																

Bits	Name		Type	Reset	Description											
31:20	RSVD															

Bits	Name	Type	Reset	Description
19:0	cr_ocram_ret	r/w	20'h0	[19] : 304 320KB OCRAM RET [18] : 288 304KB OCRAM RET [17] : 272 288KB OCRAM RET [16] : 256 272KB OCRAM RET [15] : 240 256KB OCRAM RET [14] : 224 240KB OCRAM RET [13] : 208 224KB OCRAM RET [12] : 192 208KB OCRAM RET [11] : 176 192KB OCRAM RET [10] : 160 176KB OCRAM RET [9] : 144 160KB OCRAM RET [8] : 128 144KB OCRAM RET [7] : 112 128KB OCRAM RET [6] : 96 112KB OCRAM RET [5] : 80 96KB OCRAM RET [4] : 64 80KB OCRAM RET [3] : 48 64KB OCRAM RET [2] : 32 48KB OCRAM RET [1] : 16 32KB OCRAM RET [0] : 0 16KB OCRAM RET

25.4.16 PDS_RAM4

Address: 0x2000e04c



Bits	Name	Type	Reset	Description
31:20	RSVD			

Bits	Name	Type	Reset	Description
19:0	cr_ocram_slp	r/w	20'h0	[19] : 304 320KB OCRAM SLEEP [18] : 288 304KB OCRAM SLEEP [17] : 272 288KB OCRAM SLEEP [16] : 256 272KB OCRAM SLEEP [15] : 240 256KB OCRAM SLEEP [14] : 224 240KB OCRAM SLEEP [13] : 208 224KB OCRAM SLEEP [12] : 192 208KB OCRAM SLEEP [11] : 176 192KB OCRAM SLEEP [10] : 160 176KB OCRAM SLEEP [9] : 144 160KB OCRAM SLEEP [8] : 128 144KB OCRAM SLEEP [7] : 112 128KB OCRAM SLEEP [6] : 96 112KB OCRAM SLEEP [5] : 80 96KB OCRAM SLEEP [4] : 64 80KB OCRAM SLEEP [3] : 48 64KB OCRAM SLEEP [2] : 32 48KB OCRAM SLEEP [1] : 16 32KB OCRAM SLEEP [0] : 0 16KB OCRAM SLEEP

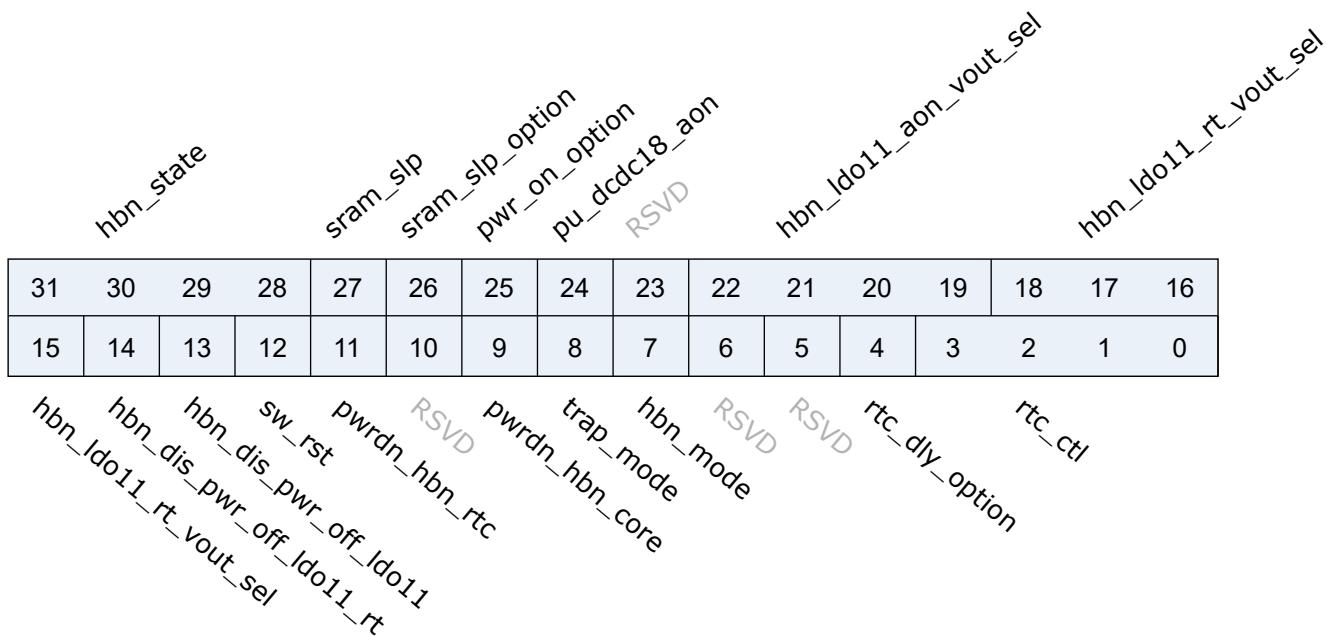
25.5 Register description

Name	Description
HBN_CTL	
HBN_TIME_L	
HBN_TIME_H	
RTC_TIME_L	
RTC_TIME_H	
HBN_IRQ_MODE	
HBN_IRQ_STAT	
HBN_IRQ_CLR	
HBN_PIR_CFG	
HBN_PIR_VTH	
HBN_PIR_INTERVAL	

Name	Description
HBN_BOR_CFG	
HBN_GLB	
HBN_SRAM	
HBN_PAD_CTRL_0	
HBN_PAD_CTRL_1	
vbat_ldo	
rc32k_ctrl0	
xtal32k	

25.5.1 HBN_CTL

Address: 0x2000f000



Bits	Name	Type	Reset	Description
31:28	hbn_state	r	4'h0	SW polling until 0 (default 4'h3 @pwron)
27	sram_slp	r	1'b0	SW polling until 0 (default 1'b1 @pwron)
26	sram_slp_option	r/w	0	
25	pwr_on_option	r/w	0	

Bits	Name	Type	Reset	Description
24	pu_dcdc18_aon	r/w	1	Power On DCDC18 during Power On Sequence (require 400 800us)
23	RSVD			
22:19	hbn_ldo11_aon_vout_sel	r/w	4'hA	VDD11_AON Voltage Out Select @ Enter hibernate
18:15	hbn_ldo11_rt_vout_sel	r/w	4'hA	VDD11_RT Voltage Out Select @ Enter hibernate
14	hbn_dis_pwr_off_ldo11_rt	r/w	0	Set 1 to disable power off VDDCORE_RT at HBN mode (for low power)
13	hbn_dis_pwr_off_ldo11	r/w	0	Set 1 to disable power off VDDCORE at HBN mode (for debug)
12	sw_RST	r/w	0	soft reset
11	pwrdown_hbn_RTC	r/w	0	Power Off HBN RTC @ Enter hibernate
10	RSVD			
9	pwrdown_hbn_core	r/w	0	Power Off HBN Core @ Enter hibernate
8	trap_mode	r	0	Boot Strap 0: Flash, 1: UART or SDIO
7	hbn_mode	w	0	Enter hibernate
6:5	RSVD			
4	rtc_dly_option	r/w	0	
3:0	rtc_ctl	r/w	4'h0	[6:4] Slow LED, x/0.25/0.5/1/2/4/8/16 seconds (move to 0x38) [3] rtc long time 0 353days (bit 39 13 compare) [2] rtc short time 0 488s (bit 23 0 compare) [1] rtc time 0 353days (bit 39 0 compare) [0] rtc enable

25.5.2 HBN_TIME_L

Address: 0x2000f004

hbn_time_l

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

hbn_time_l

Bits	Name	Type	Reset	Description
31:0	hbn_time_l	r/w	32'h0	RTC timer compare bit 31:0

25.5.3 HBN_TIME_H

Address: 0x2000f008

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

hbn_time_h

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	hbn_time_h	r/w	8'h0	RTC timer compare bit 39:32

25.5.4 RTC_TIME_L

Address: 0x2000f00c

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

rtc_time_latch_l

Bits	Name	Type	Reset	Description
31:0	rtc_time_latch_l	r	32'h0	RTC time latched value bit 31:0

25.5.5 RTC_TIME_H

Address: 0x2000f010

rtc_time_latch																
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	
RSVD																
rtc_time_latch_h																

Bits	Name		Type	Reset	Description											
31	rtc_time_latch		w	0	RTC time latch for SW read											
30:8	RSVD															
7:0	rtc_time_latch_h		r	8'h0	RTC time latched value bit 39:32											

25.5.6 HBN_IRQ_MODE

Address: 0x2000f014

pin_wakeup_en																
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	
RSVD																
pin_wakeup_sel																
irq_acomp1_en																
irq_acomp0_en																
RSVD irq_bor_en RSVD reg_en_hw_pu_pd																
hbn_pin_wakeup_mode hbn_pin_wakeup_mask																

Bits	Name		Type	Reset	Description											
31:28	RSVD															
27	pin_wakeup_en		r/w	0	Pin wakeup delay enable											
26:24	pin_wakeup_sel		r/w	3'd3	Pin wakeup delay 1 7 sec											
23:22	irq_acomp1_en		r/w	0	enable acomp1 interrupt [20] posedge [21] negedge											
21:20	irq_acomp0_en		r/w	0	enable acomp0 interrupt [20] posedge [21] negedge											

Bits	Name	Type	Reset	Description
19	RSVD			
18	irq_bor_en	r/w	0	enable brown-out interrupt
17	RSVD			
16	reg_en_hw_pu_pd	r/w	1	1: Pull GPIO17 @ pwr_on and pwr_RST 0 : no pull
15:8	RSVD			
7:4	hbn_pin_wakeup_mask	r/w	4'b0	mask hbn_pin_wakeup_event
3:0	hbn_pin_wakeup_mode	r/w	4'b0101	hbn_pin_wakeup mode 0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger

25.5.7 HBN_IRQ_STAT

Address: 0x2000f018

irq_stat

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

irq_stat

Bits	Name	Type	Reset	Description
31:0	irq_stat	r	0	[22] acomp1 [20] acomp0 [18] brown-out [17] irq_pir state [16] irq_RTC state [3:0] hbn_pin_wakeup state (GPIO19/18/17/16)

25.5.8 HBN_IRQ_CLR

Address: 0x2000f01c

irq_clr

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

irq_clr

Bits	Name	Type	Reset	Description
31:0	irq_clr	w	0	[22] irq_acomp1 clear [20] irq_acomp0 clear [18] irq_bor clear [17] irq_pir clear [16] irq_RTC clear [3:0] hbn_pin_wakeup state (GPIO19/18/17/16)

25.5.9 HBN_PIR_CFG

Address: 0x2000f020

RSVD																
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	

RSVD	gpadc_cs	pir_en	RSVD	pir_dis	RSVD	pir_lpf_sel	pir_hpf_sel									
------	------	------	------	------	------	------	----------	--------	------	---------	------	-------------	-------------	--	--	--

Bits	Name	Type	Reset	Description
31:9	RSVD			
8	gpadc_cs	r/w	0	GPADC clock source select signal 0: 32MHz clock 1: PIR clock (f32k_clk)
7	pir_en	r/w	0	pir enable
6	RSVD			
5:4	pir_dis	r/w	0	pir disable [4] low -> high won't trigger interrupt [5] high -> low won't trigger interrupt

Bits	Name	Type	Reset	Description
3	RSVD			
2	pir_lpf_sel	r/w	0	0: /1. 1:/2
1:0	pir_hpf_sel	r/w	0	0: 1-z-1, 1: 1-z-2, 0: 2-z-3

25.5.10 HBN_PIR_VTH

Address: 0x2000f024

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | |

pir_vth

Bits	Name	Type	Reset	Description
31:14	RSVD			
13:0	pir_vth	r/w	14'h3ff	PIR compare threshold

25.5.11 HBN_PIR_INTERVAL

Address: 0x2000f028

| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | | |

pir_interval

Bits	Name	Type	Reset	Description
31:12	RSVD			

Bits	Name	Type	Reset	Description
11:0	pir_interval	r/w	12'd2621	pir_lpf_sel = 0: 32768 / (pir_interval+1) Hz, default 12.5Hz (80ms) pir_lpf_sel = 1: 32768 / (pir_interval*2+1) Hz, default 6.25Hz (160ms)

25.5.12 HBN_BOR_CFG

Address: 0x2000f02c

RSVD																
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	

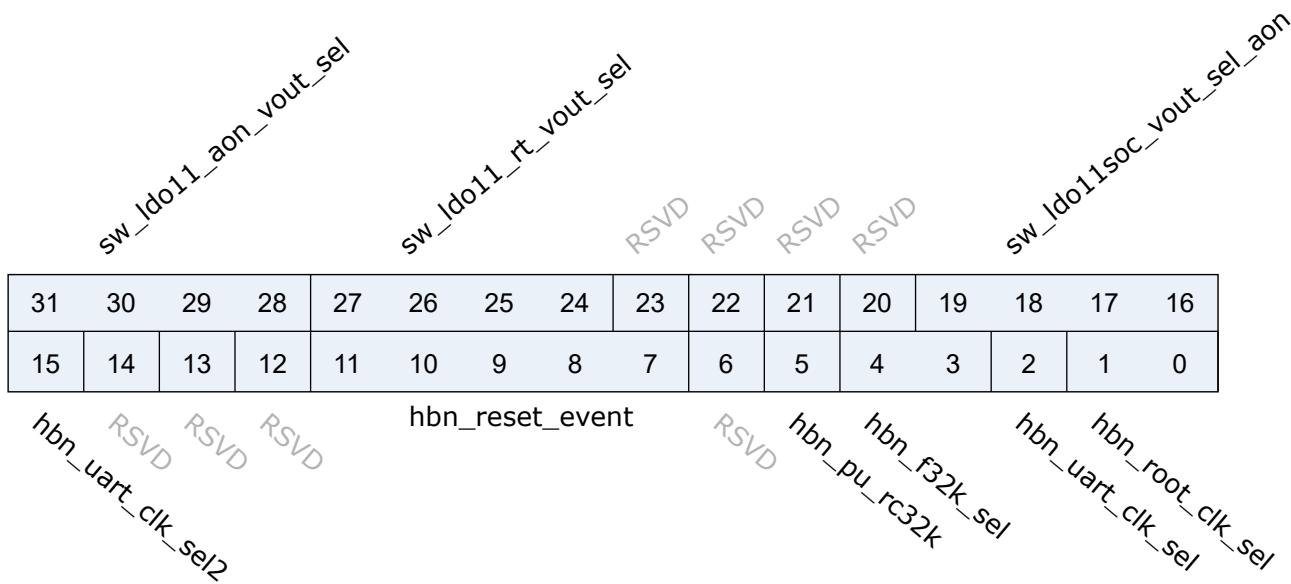
RSVD *RSVD*

r_bod_out *pu_bod* *bod_vth* *bod_sel*

Bits	Name	Type	Reset	Description
31:6	RSVD			
5	r_bod_out	r	1'h0	
4	pu_bod	r/w	1'h0	Power up Brown Out Reset
3:1	bod_vth	r/w	3'h5	bod threshold 000: 2.05V, 001: 2.10V, 010: 2.15V, 011: 2.20V, 100: 2.25V, 101: 2.30V, 110: 2.35V, 111: 2.40V
0	bod_sel	r/w	1'h0	0: POR is independent of BOD, 1: POR is relevant to BOD

25.5.13 HBN_GLB

Address: 0x2000f030



Bits	Name	Type	Reset	Description
31:28	sw_ldo11_aon_vout_sel	r/w	4'ha	aon ldo output voltage external control: 0:0.70V, 1:0.70V, 2:0.70V, 3:0.75V, 4:0.80V, 5:0.85V, 6:0.9V, 7:0.95V 8:1.0V, 9:1.05V, 10:1.1V, 11:1.15V, 12:1.2V, 13:1.25V, 14:1.3V, 15:1.35V
27:24	sw_ldo11_rt_vout_sel	r/w	4'ha	ldo output voltage external control: 0:0.70V, 1:0.70V, 2:0.70V, 3:0.75V, 4:0.80V, 5:0.85V, 6:0.9V, 7:0.95V 8:1.0V, 9:1.05V, 10:1.1V, 11:1.15V, 12:1.2V, 13:1.25V, 14:1.3V, 15:1.35V
23:20	RSVD			
19:16	sw_ldo11soc_vout_sel_aon	r/w	4'hA	vdd11soc output voltage selection
15	hbn_uart_clk_sel2	r/w	0	UART clock selection2 from HBN (0 : result of hbn_uart_clk_sel (bclk or MUX 160MHz), 1: XCLK (XTAL or RC32M))
14:12	RSVD			
11:7	hbn_reset_event	r	5'b0	[4] : bor_out_event [3] : pwr_RST_N event [2] : sw_RST event [1] : POR_OUT event [0] : watch dog reset

Bits	Name	Type	Reset	Description
6	RSVD			
5	hbn_pu_rc32k	r/w	1	0: Turn off rc32k during rtc power domain off, 1: Don't turn off rc32k (move to RTC Domain)
4:3	hbn_f32k_sel	r/w	0	32KHz clock source selection (0: RC32K 1: XTAL 32K 3: DIG 32K)
2	hbn_uart_clk_sel	r/w	0	UART clock selection from HBN (0:bclk 1:muxpll_160m_clk)
1:0	hbn_root_clk_sel	r/w	0	root clock source selection (0: RC32M 1: XTAL 2/3: PLL)

25.5.14 HBN_SRAM

Address: 0x2000f034

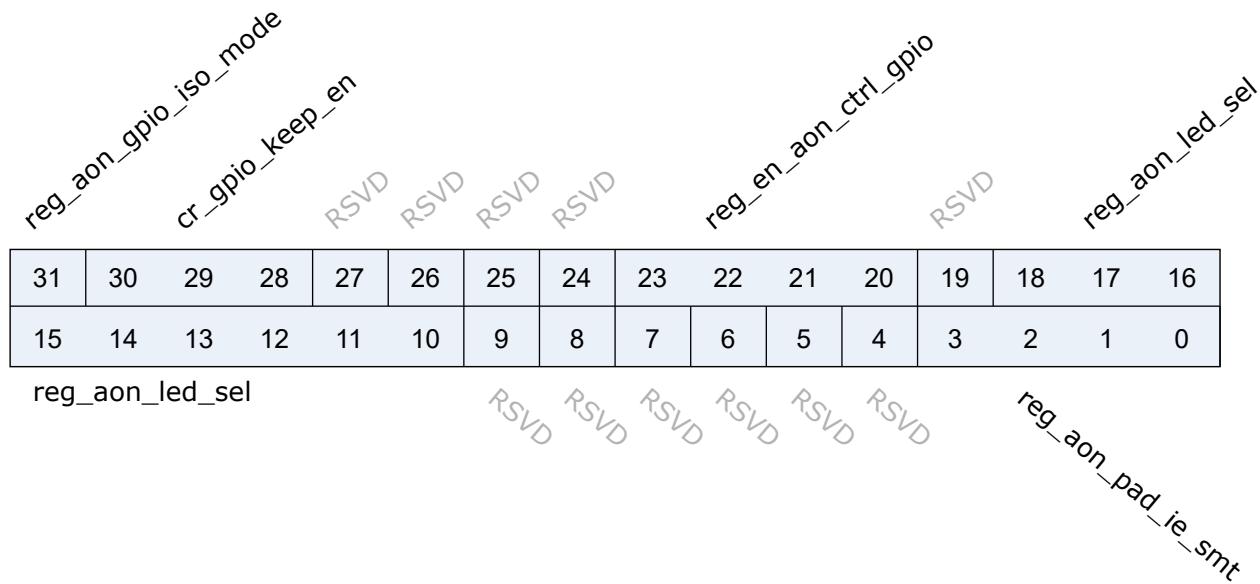
| RSVD |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
|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

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD retram_slp retram_ret RSVD RSVD RSVD RSVD RSVD RSVD

Bits	Name	Type	Reset	Description
31:8	RSVD			
7	retram_slp	r/w	0	
6	retram_ret	r/w	0	
5:0	RSVD			

25.5.15 HBN_PAD_CTRL_0

Address: 0x2000f038

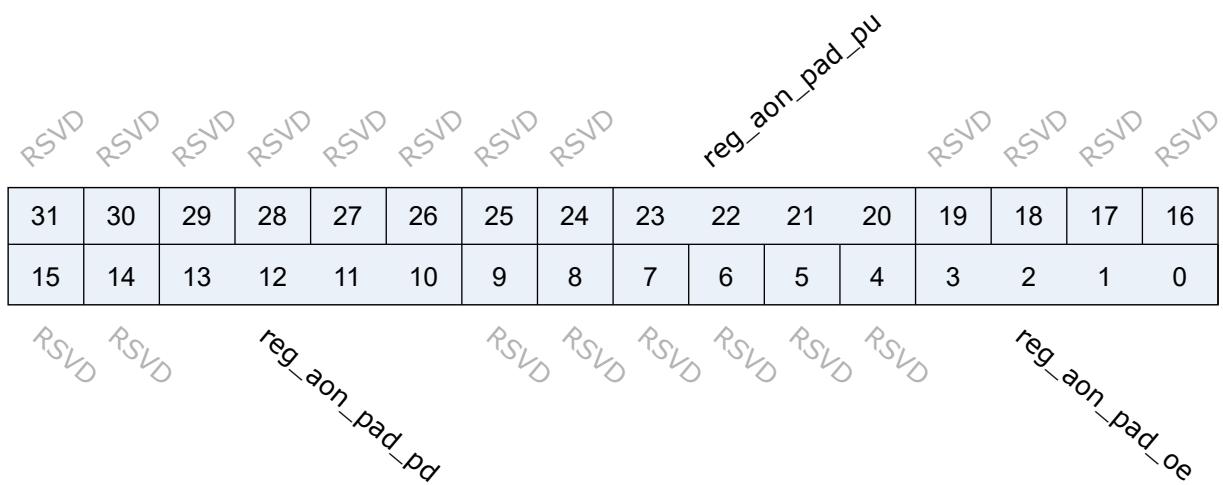


Bits	Name	Type	Reset	Description
31	reg_aon_gpio_iso_mode	r/w	0	1: HW Keep GPIO @ PDS or HBN Mode 0 : No Keep GPIO @ PDS or HBN Mode
30:28	cr_gpio_keep_en	r/w	0	if cr_aon_ctrl_gpio_latch=1, can use bit to enable or disable IO latch function at enter HBN Mode [0] : GPIO0 15 [1] : GPIO20 36 [2] : GPIO16 19
27:24	RSVD			
23:20	reg_en_aon_ctrl_gpio		4'h0	AON GPIO16 19 Control by AON HW : [23] :GPIO19 [22]: GPIO18 [21]: GPIO17(can be muxed to be XTAL32K) [20]: GPIO16(can be muxed to be XTAL32K)
19	RSVD			

Bits	Name	Type	Reset	Description
18:10	reg_aon_led_sel		9'h000	(if corresponding AON GPIO controlled by AON HW) Always on PAD Output Slow LED, x/0.25/0.5/1/2/4/8/16 seconds [12:10] (for GPIO16) : reg_aon_led1_sel [15:13] (for GPIO17) : reg_aon_led2_sel [18:16] (for GPIO18/19) : reg_aon_led3_sel 1 : 0.25sec 2 : 0.5sec 3: 1 sec 4: 2sec 5: 4 sec 6 : 8sec 7 :16sec
9:4	RSVD			
3:0	reg_aon_pad_ie_smt		4'h0	Always on PAD IE/SMT (if corresponding AON GPIO controlled by AON HW) [3] : GPIO19 [2] : GPIO18 [1] : GPIO17 [0] : GPIO16

25.5.16 HBN_PAD_CTRL_1

Address: 0x2000f03c

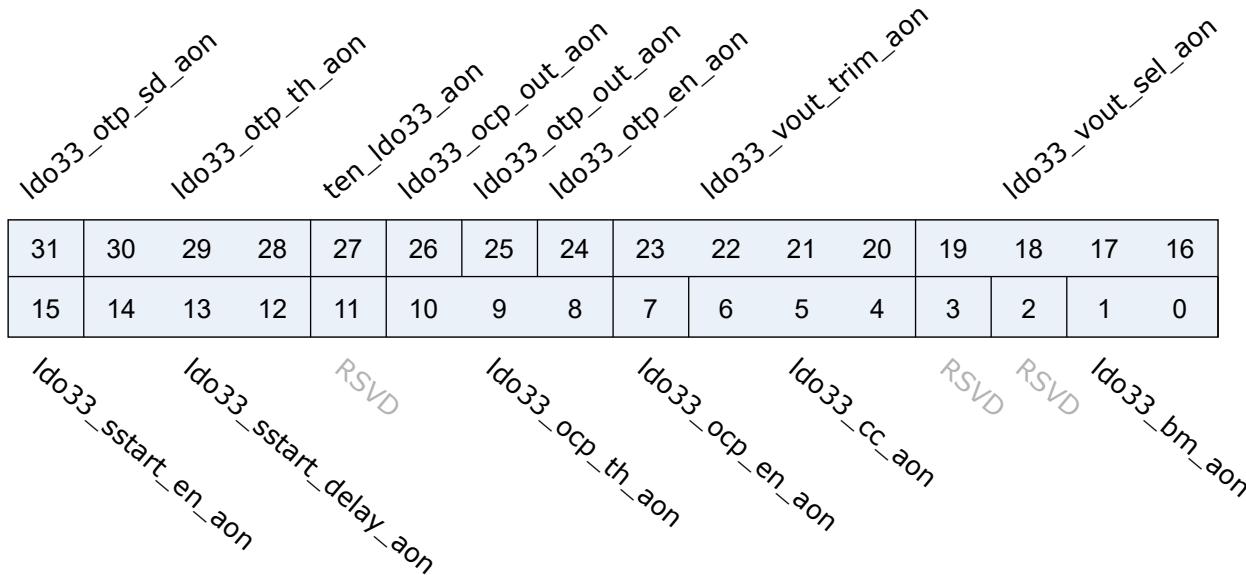


Bits	Name	Type	Reset	Description
31:24	RSVD			

Bits	Name	Type	Reset	Description
23:20	reg_aon_pad_pu		4'h0	Always on PAD PU (if corresponding AON GPIO controlled by AON HW) [23] : GPIO19 [22] : GPIO18 [21] : GPIO17 [20] : GPIO16
19:14	RSVD			
13:10	reg_aon_pad_pd		4'h0	Always on PAD PD (if corresponding AON GPIO controlled by AON HW) [13] : GPIO19 [12] : GPIO18 [11] : GPIO17 [10] : GPIO16
9:4	RSVD			
3:0	reg_aon_pad_oe		4'h0	Always on PAD OE (if corresponding AON GPIO controlled by AON HW) [3] : GPIO19 [2] : GPIO18 [1] : GPIO17 [0] : GPIO16

25.5.17 vbat_ldo

Address: 0x2000f040

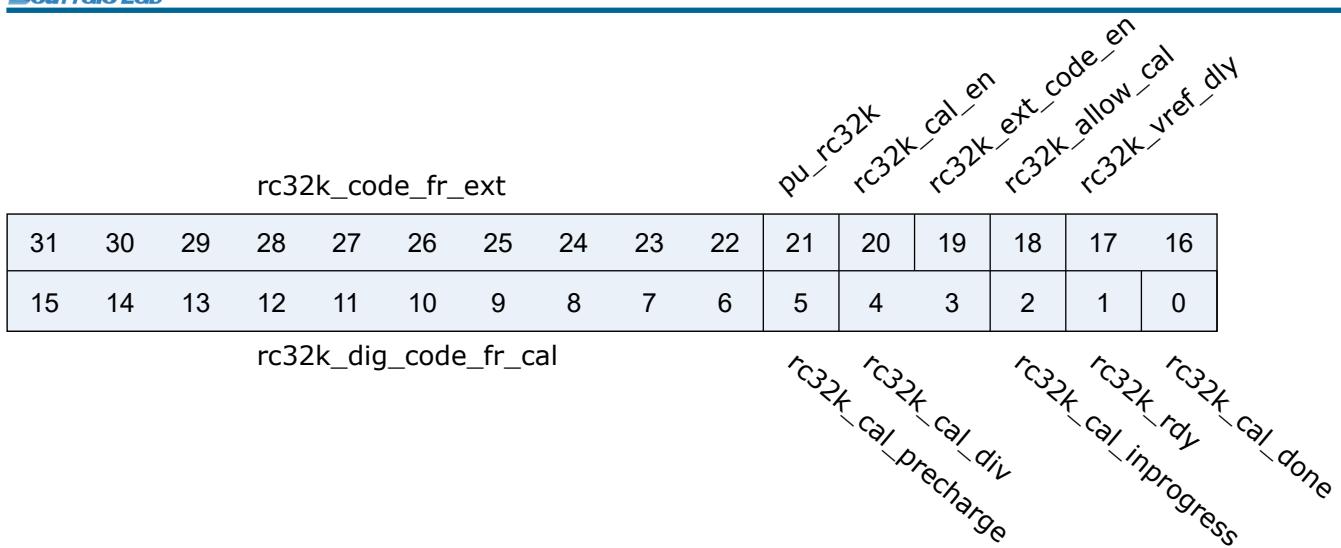


Bits	Name	Type	Reset	Description
31	ldo33_otp_sd_aon	r/w	1'h0	0: do not pulldown ldo33 when OTP happens, 1: pulldown ldo33 when OTP happens
30:28	ldo33_otp_th_aon	r/w	3'h7	OTP temperature threshold selection: 0: 125C, 1: 150C, 2: 175C, 3: 200C, 4: 200C, . . . 7: 200C
27	ten_ldo33_aon	r/w	1'h0	
26	ldo33_ocp_out_aon	r	1'h0	OC signal
25	ldo33_otp_out_aon	r	1'h0	OTP signal
24	ldo33_otp_en_aon	r/w	1'h1	enable Over Temperature Protection circuit in vbat_top
23:20	ldo33_vout_trim_aon	r/w	4'h8	output voltage trim: 1
19:16	ldo33_vout_sel_aon	r/w	4'hb	avdd33_out voltage selection: 0: 2.10, 1: 2.20, 2: 2.30, 3: 2.40, 4: 2.50, 5: 2.70, 6: 2.90, 7: 3.00, 8: 3.10, 9: 3.20, a: 3.25, b: 3.30, c: 3.35, d: 3.40, e: 3.50, f: 3.60
15	ldo33_sstart_en_aon	r/w	1'h1	enable Soft-start circuit in vbat_top

Bits	Name	Type	Reset	Description
14:12	ldo33_sstart_delay_aon	r/w	3'h3	Sstart time selection: 0: 200us, 1: 290us, 2: 390us, 3: 480us, 4: 580us, 5: 670us, 6: 770us, 7: 860us
11	RSVD			
10:8	ldo33_ocp_th_aon	r/w	3'h5	OCP current threshold selection: 0: 0.5A, 1: 0.6A, 2: 0.8A, 3: 1.0A, 4: 1.2A, 5: 1.5A, 6: 2.0A, 7: 2.0A
7	ldo33_ocp_en_aon	r/w	1'h1	enable Over Current Protection circuit in vbat_top
6:4	ldo33_cc_aon	r/w	3'h1	Compensation Capacitance selection
3:2	RSVD			
1:0	ldo33_bm_aon	r/w	2'h1	Bias mode selection

25.5.18 rc32k_ctrl0

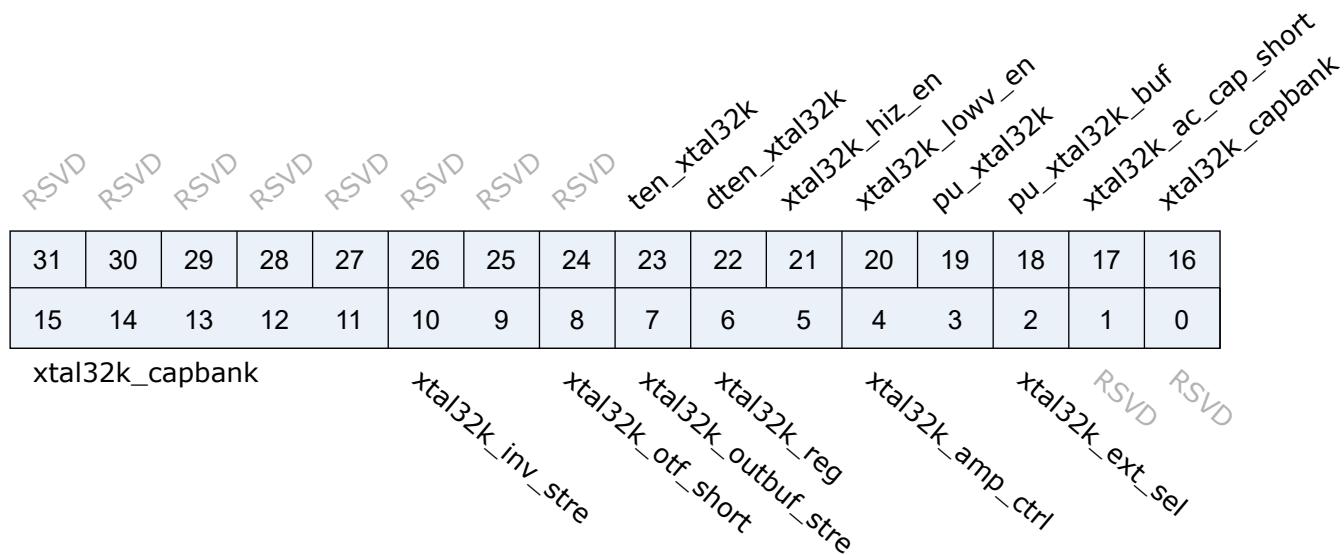
Address: 0x2000f200



Bits	Name	Type	Reset	Description
31:22	rc32k_code_fr_ext	r/w	10'h12C	External code In for frequency setting
21	pu_rc32k	r/w	1	Power up 32k oscillator (Useless in 616)
20	rc32k_cal_en	r/w	0	Enable calibration of 32k oscillator
19	rc32k_ext_code_en	r/w	1	Allow external code in to go into the ckt
18	rc32k_allow_cal	r/w	0	Allow calibration to be performed (monitor system clock)
17:16	rc32k_vref_dly	r/w	0	reference power up delay
15:6	rc32k_dig_code_fr_cal	r	10'h200	After calibration hold calibrated code
5	rc32k_cal_precharge	r	0	Initial a new cal
4:3	rc32k_cal_div	r/w	2'h3	Divider for the clock step during calibration
2	rc32k_cal_inprogress	r	0	Asserts high when calibration is in progress
1	rc32k_rdy	r	1	Asserts high when 32k clock is ready upon pwrup
0	rc32k_cal_done	r	1	Asserts high when calibration is done

25.5.19 xtal32k

Address: 0x2000f204



Bits	Name	Type	Reset	Description
31:24	RSVD			
23	ten_xtal32k	r/w	1'h0	
22	dten_xtal32k	r/w	1'h0	
21	xtal32k_hiz_en	r/w	1	high-z xtal32k input/output for share gpio usage
20	xtal32k_lowv_en	r/w	1'h0	xtal32k low voltage mode enable
19	pu_xtal32k	r/w	0	power up signal for 32K crystal oscillator
18	pu_xtal32k_buf	r/w	1	1: power up XTAL32k level shifter buffer
17	xtal32k_ac_cap_short	r/w	0	
16:11	xtal32k_capbank	r/w	6'h20	32K crystal load cap control word (also copy from efuse)
10:9	xtal32k_inv_stre	r/w	2'h1	32K crystal inverter amplify strength
8	xtal32k_otf_short	r/w	0	32K crystal over tone filter short
7	xtal32k_outbuf_stre	r/w	0	32K crystal output buffer strength
6:5	xtal32k_reg	r/w	2'h1	32K crystal voltage regulator level
4:3	xtal32k_amp_ctrl	r/w	2'h1	32K crystal oscillation amplitude control
2	xtal32k_ext_sel	r/w	0	External 32K clock enable
1:0	RSVD			

26.1 Overview

26.1.1 AES

Advanced Encryption Standard (AES) is the most common type of symmetric encryption algorithm. Also known as Rijndael encryption algorithm in cryptography, it is a block encryption standard adopted by the U.S. federal government.

26.1.2 SHA

SHA256 is a variant of Secure Hash Algorithm 2 (SHA2). SHA2, the successor of SHA1, is a cryptographic hash function algorithm standard designed by the United States National Security Agency. SHA2 has six different variants: SHA224, SHA256, SHA384, SHA512, SHA512/224, and SHA512/256. SHA-512/224 means that the result takes the first 224 bits of SHA-512, and SHA-512/256 means that the result takes the first 256 bits of SHA-512.

26.1.3 CRC

Cyclic Redundancy Check (CRC) is a channel coding technique that generates short fixed-digit check codes based on data including network packets or computer files. It is mainly used to detect or check possible errors after data transfer or storage. It detects errors based on the principle of division and remainder.

26.1.4 GMAC

GMAC is to use Galois Field (GF, finite field) multiplication to calculate the MAC value of a message.

26.2 Features

- Support AES-128, AES-192, AES-256 encryption and decryption
- Support SHA-256, SHA-512
- Support CRC-16, CRC-32
- Support GMAC
- Support True Random Number Generator(TRNG)

26.3 Principle

26.3.1 AES encryption and decryption process

AES means that the same key is used for encryption and decryption, with encryption process as follows:

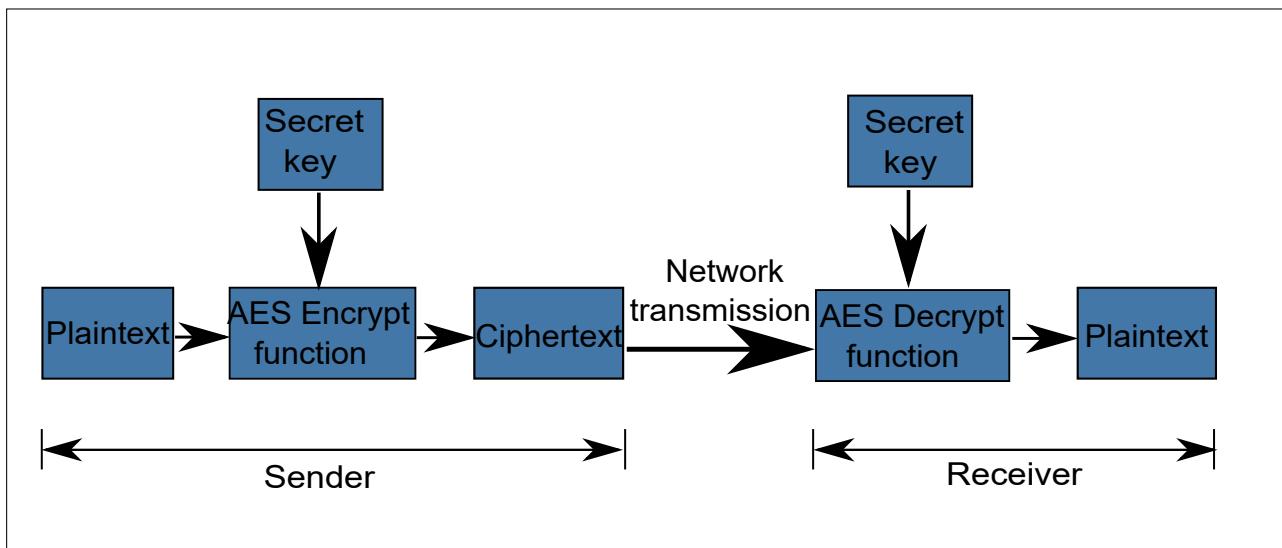


Fig. 26.1: AES Encryption Process

The following briefs the functions and significance of each part:

- Plaintext (P): unencrypted data.
- Key (K): The key used to encrypt plaintext. In symmetric encryption algorithm, encryption and decryption share one key. The key is generated by negotiation between the receiver and the sender, but it cannot be directly transmitted over network. Otherwise it will leak the key. Usually, the key is encrypted by an asymmetric encryption algorithm, and then transmitted to the other side over network, or the key is discussed face to face. The key must not be leaked. Otherwise the attacker will restore the ciphertext and steal confidential data.
- AES encryption function: Let this function be E, then $C = E(K, P)$, where P, K and C denote plaintext, key, and ciphertext, respectively. That is, if plaintext (P) and key (K) are input as the parameters of function, this function will output ciphertext (C).

- Ciphertext (C): data processed by encryption function.
- AES decryption function: Let this function be D, then $P = D(K, C)$, where C, K, and P denote ciphertext, key, and plaintext, respectively. That is, if ciphertext (C) and key (K) are input as the parameters of function, this function will output plaintext (P).

26.3.2 Implementation of SHA256

Its rule of data padding is the same as that of MD5. SHA256 uses 8 initial hash values and 64 hash constants.

The 8 initial hash values are as follows:

- $h_0 = 0x6a09e667$
- $h_1 = 0xbb67ae85$
- $h_2 = 0x3c6ef372$
- $h_3 = 0xa54ff53a$
- $h_4 = 0x510e527f$
- $h_5 = 0x9b05688c$
- $h_6 = 0x1f83d9ab$
- $h_7 = 0x5be0cd19$

These initial values are obtained by taking the first 32 bits from the decimal part of the square root of the first 8 prime numbers (2, 3, 5, 7, 11, 13, 17, and 19) in natural numbers.

The 64 constants are as follows:

- 428a2f98 71374491 b5c0fbcf e9b5dba5
- 3956c25b 59f111f1 923f82a4 ab1c5ed5
- d807aa98 12835b01 243185be 550c7dc3
- 72be5d74 80deb1fe 9bdc06a7 c19bf174
- e49b69c1 efbe4786 0fc19dc6 240ca1cc
- 2de92c6f 4a7484aa 5cb0a9dc 76f988da
- 983e5152 a831c66d b00327c8 bf597fc7
- c6e00bf3 d5a79147 06ca6351 14292967
- 27b70a85 2e1b2138 4d2c6dfc 53380d13
- 650a7354 766a0abb 81c2c92e 92722c85

- a2bfe8a1 a81a664b c24b8b70 c76c51a3
- d192e819 d6990624 f40e3585 106aa070
- 19a4c116 1e376c08 2748774c 34b0bcb5
- 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
- 748f82ee 78a5636f 84c87814 8cc70208
- 90beffa a4506ceb bef9a3f7 c67178f2

Similarly, these constants are obtained by taking the first 32 bits from the decimal part of the cube root of the first 64 prime numbers (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97...) in natural numbers.

SHA256 hash function:

- $Ch(x,y,z) = (x \square y) \square (\neg x \square z)$
- $Ma(x,y,z) = (x \square y) \square (x \square z) \square (y \square z)$
- $\Sigma_0(x) = S^2(x) \square S^{13}(x) \square S^{22}(x)$
- $\Sigma_1(x) = S^6(x) \square S^{11}(x) \square S^{25}(x)$
- $\sigma_0(x) = S^7(x) \square S^{18}(x) \square R^3(x)$
- $\sigma_1(x) = S^{17}(x) \square S^{19}(x) \square R^{10}(x)$
- \square : bitwise “AND”
- \neg : bitwise “Padding”
- \square : bitwise “Exclusive OR”
- S^n : Circularly shift right by n bits
- R^n : Shift right by n bits

26.3.3 Principle of GMAC

Authentication is actually a redundant message generated against the message itself, that is, the message authentication code (MAC). MAC is a technique for authenticating the integrity of a message. In cryptography, MAC refers to a verification mechanism used by both communication entities and a tool to ensure the integrity of message data. MAC is a hash function with a key. But why does it need a key? The reason is that the message can be tampered with during transmission, so can the hash value. Therefore, to ensure a valid hash value, the hash value is protected by encryption, so that the receiver can judge the integrity of the whole message through the hash value upon reception, thus completing information transfer.

The MAC process is shown as follows:

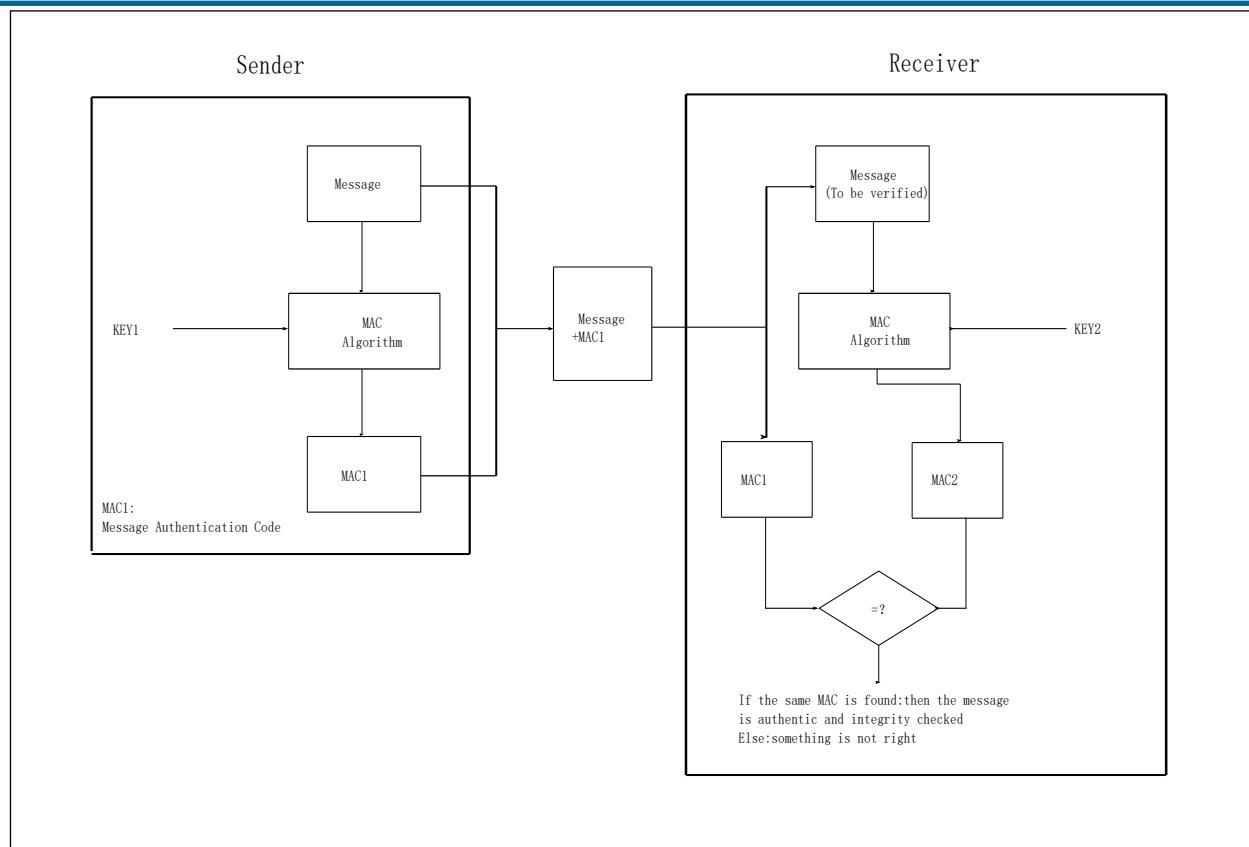


Fig. 26.2: MAC Flow Chart

1. The sender and the receiver share the key K in advance (keep KEY1 and KEY2 values in the above figure consistent).
2. The sender calculates the MAC value based on the message (KEY1 is used to calculate the MAC1 for the original message).
3. The sender sends the original message and MAC1 to the receiver.
4. The receiver calculates MAC2 using KEY2 based on the received original message .
5. The receiver compares the calculated MAC2 with the MAC1 received from the sender.
6. If the MAC is consistent, the receiver can judge that the message is indeed from the sender (authentication succeeded) and has not been tampered with or there is a transmission error. If not, the message is not from the sender (authentication failed).

Note: It is recommended that the sender and the receiver store the KEY in the hardware security module, and the MAC value also shall be calculated in that module, to ensure the security of KEY, for example, in the encrypted chip.

GMAC is to use Galois Field (GF, finite field) multiplication to calculate the MAC value of a message.

26.4 Functional Description

26.4.1 AES Accelerator

1. AES accelerator supports AES128/192/256 encryption and decryption.
- Configure se_aes_0_mode and se_aes_0_dec_en in the register se_aes_0_ctrlL, as shown below:

amode	adec en	运算
0	0	AES-128 加密
1	0	AES-256 加密
2	0	AES-192 加密
0	1	AES-128 解密
1	1	AES-256 解密
2	1	AES-192 解密

Fig. 26.3: AES Operation Modes

Configure se_aes_0_block_mode in the register se_aes_0_ctrlL to select among different encryption modes including ECB, CTR, CBC, and XTS.

2. Key, plaintext, ciphertext, and initialization vector
 - The register se_aes_0_msa stores the address of plaintext or ciphertext.
 - The register se_aes_0_msa stores the address of ciphertext or plaintext.
 - The register se_aes_0_iv_0~se_aes_0_iv_3 stores IV.
 - The register se_aes_0_key_0~se_aes_0_key_7 stores the key.
3. Software and hardware encryption process
 - Configure the register se_aes_0_endian, including se_aes_0_dout_endian, se_aes_0_din_endian, se_aes_0_key_endian, se_aes_0_iv_endian, and se_aes_0_twk_endian; the value of 0 indicates “little endian” and the value of 1 indicates “big endian”
 - Configure se_aes_0_block_mode in the register se_aes_0_ctrl
 - Configure se_aes_0_mode in the register se_aes_0_ctrl
 - Configure se_aes_0_dec_en in the register se_aes_0_ctrl
 - Configure se_aes_0_dec_key_sel in the register se_aes_0_ctrl; 0 means using a new key, and 1 means using the

same key as the last time

- Configure se_aes_0_iv_sel in the register se_aes_0_ctrl; 0 means using a new iv, and 1 means using the same iv as the last time
- Enable AES by configuring se_aes_0_en in the register se_aes_0_ctrl
- Configure the register se_aes_0_iv_0~se_aes_0_iv_3 to set IV; the sequence of filling for MSB is se_aes_0_iv_0~se_aes_0_iv_3 and that for LSB is se_aes_0_iv_3~se_aes_0_iv_0
- Configure the register se_aes_0_key_0~se_aes_0_key_7 to set key; the sequence of filling for MSB is se_aes_0_key_0~se_aes_0_key_7 and that for LSB is se_aes_0_key_7~se_aes_0_key_0. Take first 4 bits for AES128, first 6 bits for AES196, and first 8 bits for AES256
- Configure the register se_aes_0_msa to set the source address of the data to be processed
- Configure the register se_aes_0_mda to set the destination address where the processing results are stored
- Configure se_aes_0_msg_len in the register se_aes_0_ctrl to set the length of the data to be processed, in units of 128 bit
- Configure se_aes_0_trig_1t in the register se_aes_0_ctrl to trigger AES
- The output result is stored in the address corresponding to the register se_aes_0_mda

26.4.2 SHA Accelerator

26.4.3 SHA Accelerator

1. SHA accelerator supports 7 standard operations:

- SHA-1、SHA-224、SHA-256、SHA-512、SHA-384、SHA-512/224、SHA-512/256，同时还支持 MD5、CRC16、CRC32。

The se_sha_0_mode in the register se_sha_0_ctrl: 0:SHA-256 1:SHA-224 2:SHA-1 3:SHA-1 4:SHA-512 5:SHA-384 6:SHA-512/224 7:SHA-512/256

The se_sha_0_mode_ext in the register se_sha_0_ctrl: hash mode extention; 0:SHA 1:MD5 2:CRC-16 3:CRC-32

Configure se_sha_0_mode in the register se_sha_0_ctrl to select among different SHA operations, and configure se_sha_0_mode_ext in the register se_sha_0_ctrl to select among MD5, CRC16, and CRC32

- The se_sha_0_mode is valid when se_sha_0_mode_ext is 0.
- The se_sha_0_mode is invalid when se_sha_0_mode_ext is not 0.

2. Plaintext and ciphertext

- The register se_sha_0_msa stores the plaintext address.
- The register se_sha_0_hash_l_0~se_sha_0_hash_l_7 stores the ciphertext.

Sequence: MSB: se_sha_0_hash_l_0~se_sha_0_hash_l_7; LSB: se_sha_0_hash_l_7~se_sha_0_hash_l_0

3. Operation flow

- Configure se_sha_0_mode in the register se_sha_0_ctrl to set the specific mode of SHA
- Enable SHA by configuring se_sha_0_en in the register se_sha_0_ctrl
- Configure se_sha_0_hash_sel in the register se_sha_0_ctrl; 0 means starting a new HASH calculation, and 1 means using the last result for HASH calculation
- Configure the register se_sha_0_msa to set the source address of the data to be processed
- Configure se_sha_0_msg_len in the register se_sha_0_ctrl to set the length of the data to be processed (512 bits for SHA1, SHA224 and SHA256, while 1024 bits for SHA512, SHA384, SHA512/224, and SHA512/256)
- Configure se_sha_0_trig_1t in the register se_sha_0_ctrl to trigger SHA
- The output result is stored in se_sha_0_hash_l_0~se_sha_0_hash_l_7, MSB:se_sha_0_hash_l_0~se_sha_0_hash_l_7, LSB:se_sha_0_hash_l_7~se_sha_0_hash_l_0

26.4.4 GMAC(link Mode)

1. Definition of GMAC_link_Table Structure

- Word0:
 - [9]:se_gmac_0_int_clr_1t
 - [10]:se_gmac_0_int_set_1t
 - [31:16]:se_gmac_0_msg_len
- Word1:se_gmac_0_msa
- Word2、Word3、Word4、Word5: se_gmac_0_h
- Word6、Word7、Word8、Word9: se_gmac_0_tag

2. Usage process

- Configure se_gmac_0_x_endian, se_gmac_0_h_endian, and se_gmac_0_t_endian 0:little endian 1:big endian in the register se_gmac_0_ctrl_0
- Write the start address of the GMAC_link_Table structure into the register se_gmac_0_lca
- Assign the original message address to se_gmac_0_msa in Word1 in GMAC_link_Table structure
- Assign the original message length to se_gmac_0_msg_len in Word0 in GMAC_link_Table structure, where the 128-bit message length corresponds to 1 in se_gmac_0_msg_len
- Configure se_gmac_0_trig_1t trigger gmac engine in the register se_gmac_0_ctrl_0

- The output result is stored in Word6–Word9 in the GMAC_link_Table structure

26.4.5 Random Number Generator (RNG)

1. The random numbers generated by the built-in true RNG can be used as the basis for encryption and other operations.
 - True random numbers: They are generated through physical phenomena, such as coin tossup, dicing, wheel rotation, noise from using electronic components, and nuclear fission. Such RNGs are called physical RNGs, and their weaknesses are high technical requirements.
 - Pseudo-random numbers: Truly random numbers (or random events) are randomly generated in a generation process according to the distribution probability shown in the experimental process, and the result is unpredictable and invisible. The random function in the computer is simulated according to a specified algorithm, and its result is deterministic and visible. We may consider that the probability of this foreseeable result is 100%. Hence the “random number” generated by computer random function is not truly random, but pseudo-random.
2. Output
 - The register SE_TRNG_0_DOUT_0~SE_TRNG_0_DOUT_7 stores the output random numbers.
 3. Usage process
 - Enable TRNG by configuring se_trng_0_en in the register se_trng_0_ctrl_0
 - Configure se_trng_0_trig_1t in the register se_trng_0_ctrl_0 to trigger TRNG
 - The output result is stored in se_trng_0_dout_0~se_trng_0_dout_7

Revision history

Table 27.1: Document revision history

Date	Revision	Changes
2021/9/22	0.9	Initial version
2022/8/22	0.91	Add register description