

# ESP32-S2

## Technical Reference Manual



Preliminary V0.5  
Espressif Systems  
Copyright © 2020

## About This Manual

The **ESP32-S2 Technical Reference Manual** is addressed to application developers. The manual provides detailed and complete information on how to use the ESP32-S2 memory and peripherals.

For pin definition, electrical characteristics and package information, please see [ESP32-S2 Datasheet](#).

## Document Updates

Please always refer to the latest version on <https://www.espressif.com/en/support/download/documents>.

## Revision History

For any changes to this document over time, please refer to the [last page](#).

## Documentation Change Notification

Espressif provides email notifications to keep customers updated on changes to technical documentation.

Please subscribe at [www.espressif.com/en/subscribe](http://www.espressif.com/en/subscribe).

## Certification

Download certificates for Espressif products from [www.espressif.com/en/certificates](http://www.espressif.com/en/certificates).

## Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice. THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

All liability, including liability for infringement of any proprietary rights, relating to the use of information in this document, is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein. The Wi-Fi Alliance Member logo is a trademark of the Wi-Fi Alliance. The Bluetooth logo is a registered trademark of Bluetooth SIG.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Copyright © 2020 Espressif Systems (Shanghai) Co., Ltd. All rights reserved.

# Contents

<b>1</b>	<b>System and Memory</b>	<b>23</b>
1.1	Overview	23
1.2	Features	23
1.3	Functional Description	24
1.3.1	Address Mapping	24
1.3.2	Internal Memory	25
1.3.2.1	Internal ROM 0	26
1.3.2.2	Internal ROM 1	26
1.3.2.3	Internal SRAM 0	26
1.3.2.4	Internal SRAM 1	26
1.3.2.5	RTC FAST Memory	27
1.3.2.6	RTC SLOW Memory	27
1.3.3	External Memory	27
1.3.3.1	External Memory Address Mapping	27
1.3.3.2	Cache	28
1.3.3.3	Cache Operations	28
1.3.4	DMA	29
1.3.5	Modules / Peripherals	29
1.3.5.1	Naming Conventions for Peripheral Buses	29
1.3.5.2	Differences Between PeriBus1 and PeriBus2	30
1.3.5.3	Module / Peripheral Address Mapping	30
1.3.5.4	Addresses with Restricted Access from PeriBus1	31
<b>2</b>	<b>Reset and Clock</b>	<b>33</b>
2.1	Reset	33
2.1.1	Overview	33
2.1.2	Reset Source	33
2.2	Clock	34
2.2.1	Overview	34
2.2.2	Clock Source	35
2.2.3	CPU Clock	35
2.2.4	Peripheral Clock	37
2.2.4.1	APB_CLK Source	37
2.2.4.2	REF_TICK Source	37
2.2.4.3	LEDC_PWM_CLK Source	38
2.2.4.4	APLL_SCLK Source	38
2.2.4.5	PLL_160M_CLK Source	38
2.2.4.6	Clock Source Considerations	38
2.2.5	Wi-Fi Clock	38
2.2.6	RTC Clock	39
2.2.7	Audio PLL Clock	39
<b>3</b>	<b>Chip Boot Control</b>	<b>40</b>

3.1	Overview . . . . .	40
3.2	Boot Mode . . . . .	40
3.3	ROM Code Printing to UART . . . . .	41
3.4	VDD_SPI Voltage . . . . .	41
<b>4</b>	<b>Interrupt Matrix . . . . .</b>	<b>42</b>
4.1	Overview . . . . .	42
4.2	Features . . . . .	42
4.3	Functional Description . . . . .	42
4.3.1	Peripheral Interrupt Sources . . . . .	42
4.3.2	CPU Interrupts . . . . .	46
4.3.3	Allocate Peripheral Interrupt Source to CPU Interrupt . . . . .	47
4.3.3.1	Allocate one peripheral interrupt source Source_X to CPU . . . . .	47
4.3.3.2	Allocate multiple peripheral interrupt sources Source_Xn to CPU . . . . .	47
4.3.3.3	Disable CPU peripheral interrupt source Source_X . . . . .	47
4.3.4	Disable CPU NMI Interrupt Sources . . . . .	48
4.3.5	Query Current Interrupt Status of Peripheral Interrupt Source . . . . .	48
4.4	Base Address . . . . .	48
4.5	Register Summary . . . . .	48
4.6	Registers . . . . .	53
<b>5</b>	<b>IO MUX and GPIO Matrix . . . . .</b>	<b>90</b>
5.1	Overview . . . . .	90
5.2	Peripheral Input via GPIO Matrix . . . . .	91
5.2.1	Overview . . . . .	91
5.2.2	Synchronization . . . . .	91
5.2.3	Functional Description . . . . .	92
5.2.4	Simple GPIO Input . . . . .	93
5.3	Peripheral Output via GPIO Matrix . . . . .	93
5.3.1	Overview . . . . .	93
5.3.2	Functional Description . . . . .	93
5.3.3	Simple GPIO Output . . . . .	94
5.3.4	Sigma Delta Modulated Output . . . . .	95
5.3.4.1	Functional Description . . . . .	95
5.3.4.2	SDM Configuration . . . . .	95
5.4	Dedicated GPIO . . . . .	96
5.4.1	Overview . . . . .	96
5.4.2	Features . . . . .	96
5.4.3	Functional Description . . . . .	97
5.4.3.1	Accessing GPIO via Registers . . . . .	97
5.4.3.2	Accessing GPIO with CPU . . . . .	97
5.5	Direct I/O via IO MUX . . . . .	98
5.5.1	Overview . . . . .	98
5.5.2	Functional Description . . . . .	99
5.6	RTC IO MUX for Low Power and Analog I/O . . . . .	99
5.6.1	Overview . . . . .	99

5.6.2	Functional Description	99
5.7	Pin Functions in Light-sleep	99
5.8	Pad Hold Feature	100
5.9	I/O Pad Power Supplies	100
5.9.1	Power Supply Management	100
5.10	Peripheral Signal List	101
5.11	IO MUX Pad List	105
5.12	RTC IO MUX Pin List	106
5.13	Base Address	107
5.14	Register Summary	107
5.14.1	GPIO Matrix Register Summary	107
5.14.2	IO MUX Register Summary	109
5.14.3	Sigma Delta Modulated Output Register Summary	110
5.14.4	Dedicated GPIO Register Summary	110
5.14.5	RTC IO MUX Register Summary	111
5.15	Registers	112
5.15.1	GPIO Matrix Registers	112
5.15.2	IO MUX Registers	124
5.15.3	Sigma Delta Modulated Output Registers	126
5.15.4	Dedicated GPIO Registers	127
5.15.5	RTC IO MUX Registers	136
<b>6</b>	<b>System Registers</b>	<b>150</b>
6.1	Overview	150
6.2	Features	150
6.3	Function Description	150
6.3.1	System and Memory Registers	150
6.3.2	Reset and Clock Registers	152
6.3.3	Interrupt Matrix Registers	152
6.3.4	JTAG Software Enable Registers	152
6.3.5	Low-power Management Registers	153
6.3.6	Peripheral Clock Gating and Reset Registers	153
6.4	Base Address	155
6.5	Register Summary	155
6.6	Registers	156
<b>7</b>	<b>DMA Controller</b>	<b>171</b>
7.1	Overview	171
7.2	Features	171
7.3	Functional Description	172
7.3.1	DMA Engine Architecture	172
7.3.2	Linked List	172
7.3.3	Enabling DMA	173
7.3.4	Linked List reading process	174
7.3.5	EOF	174
7.3.6	Internal DMA	174

7.3.7	EDMA	175
7.3.7.1	Accessing Internal RAM	175
7.3.8	Accessing External RAM	175
7.4	Copy DMA Controller	176
7.5	UART DMA (UDMA) Controller	176
7.6	SPI DMA Controller	177
7.7	I2S DMA Controller	178
7.8	Crypto DMA	179
7.9	Copy DMA Interrupts	179
7.10	Crypto DMA Interrupts	179
7.11	Base Address	180
7.12	Register Summary	180
7.13	Registers	183
<b>8</b>	<b>UART Controller</b>	<b>207</b>
8.1	Overview	207
8.2	Features	207
8.3	Functional Description	207
8.3.1	UART Introduction	207
8.3.2	UART Structure	208
8.3.3	UART RAM	209
8.3.4	Baud Rate Generation and Detection	209
8.3.4.1	Baud Rate Generation	210
8.3.4.2	Baud Rate Detection	210
8.3.5	UART Data Frame	211
8.3.6	RS485	212
8.3.6.1	Driver Control	212
8.3.6.2	Turnaround Delay	213
8.3.6.3	Bus Snooping	213
8.3.7	IrDA	213
8.3.8	Wake-up	214
8.3.9	Flow Control	214
8.3.9.1	Hardware Flow Control	215
8.3.9.2	Software Flow Control	216
8.3.10	UDMA	216
8.3.11	UART Interrupts	216
8.3.12	UHCI Interrupts	217
8.4	Base Address	218
8.5	Register Summary	218
8.6	Registers	221
<b>9</b>	<b>LED PWM Controller</b>	<b>267</b>
9.1	Overview	267
9.2	Features	267
9.3	Functional Description	267
9.3.1	Architecture	267

9.3.2	Timers . . . . .	268
9.3.3	PWM Generators . . . . .	269
9.3.4	Duty Cycle Fading . . . . .	269
9.3.5	Interrupts . . . . .	270
9.4	Base Address . . . . .	270
9.5	Register Summary . . . . .	271
9.6	Registers . . . . .	273
<b>10</b>	<b>Remote Control Peripheral . . . . .</b>	<b>280</b>
10.1	Introduction . . . . .	280
10.2	Functional Description . . . . .	280
10.2.1	RMT Architecture . . . . .	280
10.2.2	RMT RAM . . . . .	281
10.2.3	Clock . . . . .	281
10.2.4	Transmitter . . . . .	281
10.2.5	Receiver . . . . .	282
10.2.6	Interrupts . . . . .	282
10.3	Base Address . . . . .	283
10.4	Register Summary . . . . .	283
10.5	Registers . . . . .	285
<b>11</b>	<b>Pulse Count Controller . . . . .</b>	<b>294</b>
11.1	Features . . . . .	294
11.2	Functional Description . . . . .	295
11.3	Applications . . . . .	297
11.3.1	Channel 0 Incrementing Independently . . . . .	297
11.3.2	Channel 0 Decrementing Independently . . . . .	298
11.3.3	Channel 0 and Channel 1 Incrementing Together . . . . .	298
11.4	Base Address . . . . .	299
11.5	Register Summary . . . . .	299
11.6	Registers . . . . .	301
<b>12</b>	<b>64-bit Timers . . . . .</b>	<b>307</b>
12.1	Overview . . . . .	307
12.2	Functional Description . . . . .	308
12.2.1	16-bit Prescaler and Clock Selection . . . . .	308
12.2.2	64-bit Time-based Counter . . . . .	308
12.2.3	Alarm Generation . . . . .	308
12.2.4	Timer Reload . . . . .	308
12.2.5	Interrupts . . . . .	309
12.3	Configuration and Usage . . . . .	310
12.3.1	Timer as a Simple Clock . . . . .	310
12.3.2	Timer as One-shot Alarm . . . . .	310
12.3.3	Timer as Periodic Alarm . . . . .	311
12.4	Base Address . . . . .	311
12.5	Register Summary . . . . .	311

12.6	Registers	313
<b>13</b>	<b>Watchdog Timers</b>	<b>322</b>
13.1	Overview	322
13.2	Features	322
13.3	Functional Description	322
13.3.1	Clock Source and 32-Bit Counter	322
13.3.2	Stages and Timeout Actions	323
13.3.3	Write Protection	323
13.3.4	Flash Boot Protection	324
13.4	Super Watchdog	324
13.4.1	Features	324
13.4.2	Super Watchdog Controller	324
13.4.2.1	Structure	325
13.4.2.2	Workflow	325
13.5	Registers	325
<b>14</b>	<b>XTAL32K Watchdog</b>	<b>326</b>
14.1	Overview	326
14.2	Features	326
14.2.1	XTAL32K Watchdog Interrupts and Wake-up	326
14.2.2	BACKUP32K_CLK	326
14.3	Functional Description	326
14.3.1	Workflow	326
14.3.2	Configuring the Divisor of BACKUP32K_CLK	327
<b>15</b>	<b>System Timer</b>	<b>328</b>
15.1	Overview	328
15.2	Main Features	328
15.3	Clock Source Selection	328
15.4	Functional Description	328
15.4.1	Read System Timer Value	329
15.4.2	Configure a Time-Delay Alarm	329
15.4.3	Configure Periodic Alarms	329
15.4.4	Update after Deep-sleep and Light-sleep	329
15.5	Base Address	330
15.6	Register Summary	330
15.7	Registers	331
<b>16</b>	<b>eFuse Controller</b>	<b>339</b>
16.1	Overview	339
16.2	Features	339
16.3	Functional Description	339
16.3.1	Structure	339
16.3.1.1	EFUSE_WR_DIS	343
16.3.1.2	EFUSE_RD_DIS	343



16.3.1.3	Data Storage	344
16.3.2	Software Programming of Parameters	344
16.3.3	Software Reading of Parameters	346
16.3.4	Timing	347
16.3.4.1	eFuse-Programming Timing	347
16.3.4.2	eFuse VDDQ Timing Setting	348
16.3.4.3	eFuse-Read Timing	348
16.3.5	The Use of Parameters by Hardware Modules	349
16.3.6	Interrupts	349
16.4	Base Address	349
16.5	Register Summary	349
16.6	Registers	353
<b>17</b>	<b>I2C Controller</b>	<b>375</b>
17.1	Overview	375
17.2	Features	375
17.3	I2C Functional Description	375
17.3.1	I2C Introduction	375
17.3.2	I2C Architecture	376
17.3.2.1	TX/RX RAM	377
17.3.2.2	CMD_Controller	377
17.3.2.3	SCL_FSM	378
17.3.2.4	SCL_MAIN_FSM	379
17.3.2.5	DATA_Shifter	379
17.3.2.6	SCL_Filter and SDA_Filter	379
17.3.3	I2C Bus Timing	379
17.4	Typical Applications	380
17.4.1	An I2C Master Writes to an I2C Slave with a 7-bit Address in One Command Sequence	381
17.4.2	An I2C Master Writes to an I2C Slave with a 10-bit Address in One Command Sequence	382
17.4.3	An I2C Master Writes to an I2C Slave with Two 7-bit Addresses in One Command Sequence	383
17.4.4	An I2C Master Writes to an I2C Slave with a 7-bit Address in Multiple Command Sequences	383
17.4.5	An I2C Master Reads an I2C Slave with a 7-bit Address in One Command Sequence	384
17.4.6	An I2C Master Reads an I2C Slave with a 10-bit Address in One Command Sequence	385
17.4.7	An I2C Master Reads an I2C Slave with Two 7-bit Addresses in One Command Sequence	386
17.4.8	An I2C Master Reads an I2C Slave with a 7-bit Address in Multiple Command Sequences	387
17.5	Clock Stretching	387
17.6	Interrupts	388
17.7	Base Address	389
17.8	Register Summary	389
17.9	Registers	390
<b>18</b>	<b>TWAI</b>	<b>413</b>
18.1	Overview	413
18.2	Features	413
18.3	Functional Protocol	414
18.3.1	TWAI Properties	414

18.3.2	TWAI Messages	414
18.3.2.1	Data Frames and Remote Frames	415
18.3.2.2	Error and Overload Frames	417
18.3.2.3	Interframe Space	418
18.3.3	TWAI Errors	419
18.3.3.1	Error Types	419
18.3.3.2	Error States	419
18.3.3.3	Error Counters	420
18.3.4	TWAI Bit Timing	421
18.3.4.1	Nominal Bit	421
18.3.4.2	Hard Synchronization and Resynchronization	422
18.4	Architectural Overview	422
18.4.1	Registers Block	423
18.4.2	Bit Stream Processor	424
18.4.3	Error Management Logic	424
18.4.4	Bit Timing Logic	424
18.4.5	Acceptance Filter	424
18.4.6	Receive FIFO	424
18.5	Functional Description	425
18.5.1	Modes	425
18.5.1.1	Reset Mode	425
18.5.1.2	Operation Mode	425
18.5.2	Bit Timing	425
18.5.3	Interrupt Management	426
18.5.3.1	Receive Interrupt (RXI)	426
18.5.3.2	Transmit Interrupt (TXI)	427
18.5.3.3	Error Warning Interrupt (EWI)	427
18.5.3.4	Data Overrun Interrupt (DOI)	427
18.5.3.5	Error Passive Interrupt (TXI)	428
18.5.3.6	Arbitration Lost Interrupt (ALI)	428
18.5.3.7	Bus Error Interrupt (BEI)	428
18.5.4	Transmit and Receive Buffers	428
18.5.4.1	Overview of Buffers	428
18.5.4.2	Frame Information	429
18.5.4.3	Frame Identifier	429
18.5.4.4	Frame Data	430
18.5.5	Receive FIFO and Data Overruns	430
18.5.6	Acceptance Filter	431
18.5.6.1	Single Filter Mode	431
18.5.6.2	Dual Filter Mode	432
18.5.7	Error Management	433
18.5.7.1	Error Warning Limit	434
18.5.7.2	Error Passive	434
18.5.7.3	Bus-Off and Bus-Off Recovery	434
18.5.8	Error Code Capture	435
18.5.9	Arbitration Lost Capture	436

18.6	Base Address . . . . .	436
18.7	Register Summary . . . . .	437
18.8	Register Description . . . . .	438
<b>19</b>	<b>AES Accelerator . . . . .</b>	<b>451</b>
19.1	Introduction . . . . .	451
19.2	Features . . . . .	451
19.3	Working Modes . . . . .	451
19.4	Typical AES Working Mode . . . . .	452
19.4.1	Key, Plaintext, and Ciphertext . . . . .	452
19.4.2	Endianness . . . . .	453
19.4.3	Operation Process . . . . .	457
19.5	DMA-AES Working Mode . . . . .	458
19.5.1	Key, Plaintext, and Ciphertext . . . . .	458
19.5.2	Endianness . . . . .	459
19.5.3	Standard Incrementing Function . . . . .	460
19.5.4	Block Number . . . . .	460
19.5.5	Initialization Vector . . . . .	460
19.5.6	Block Operation Process . . . . .	460
19.5.7	GCM Operation Process . . . . .	461
19.6	GCM Algorithm . . . . .	462
19.6.1	Hash Subkey . . . . .	463
19.6.2	$J_0$ . . . . .	463
19.6.3	Authenticated Tag . . . . .	463
19.6.4	AAD Block Number . . . . .	463
19.6.5	Remainder Bit Number . . . . .	464
19.7	Base Address . . . . .	464
19.8	Memory Summary . . . . .	464
19.9	Register Summary . . . . .	464
19.10	Registers . . . . .	466
<b>20</b>	<b>SHA Accelerator . . . . .</b>	<b>471</b>
20.1	Introduction . . . . .	471
20.2	Features . . . . .	471
20.3	Working Modes . . . . .	471
20.4	Function Description . . . . .	472
20.4.1	Preprocessing . . . . .	472
20.4.1.1	Padding the Message . . . . .	472
20.4.1.2	Parsing the Message . . . . .	473
20.4.1.3	Initial Hash Value . . . . .	474
20.4.2	Hash Computation Process . . . . .	475
20.4.2.1	Typical SHA Process . . . . .	475
20.4.2.2	DMA-SHA Process . . . . .	477
20.4.3	Message Digest . . . . .	478
20.4.4	Interrupt . . . . .	479
20.5	Base Address . . . . .	480

20.6	Register Summary	480
20.7	Registers	482
<b>21</b>	<b>RSA Accelerator</b>	<b>486</b>
21.1	Introduction	486
21.2	Features	486
21.3	Functional Description	486
21.3.1	Large Number Modular Exponentiation	487
21.3.2	Large Number Modular Multiplication	488
21.3.3	Large Number Multiplication	489
21.3.4	Acceleration Options	489
21.4	Base Address	491
21.5	Memory Summary	491
21.6	Register Summary	491
21.7	Registers	492
<b>22</b>	<b>Random Number Generator</b>	<b>496</b>
22.1	Introduction	496
22.2	Features	496
22.3	Functional Description	496
22.4	Base Address	497
22.5	Register Summary	497
22.6	Register	497
<b>23</b>	<b>External Memory Encryption and Decryption</b>	<b>498</b>
23.1	Overview	498
23.2	Features	498
23.3	Functional Description	498
23.3.1	XTS Algorithm	499
23.3.2	Key	499
23.3.3	Target Memory Space	500
23.3.4	Data Padding	500
23.3.5	Manual Encryption Block	501
23.3.6	Auto Encryption Block	502
23.3.7	Auto Decryption Block	502
23.4	Base Address	503
23.5	Register Summary	503
23.6	Registers	504
<b>24</b>	<b>Permission Control</b>	<b>508</b>
24.1	Overview	508
24.2	Features	508
24.3	Functional Description	508
24.3.1	Internal Memory Permission Controls	508
24.3.1.1	Permission Control for the Instruction Bus (IBUS)	509
24.3.1.2	Permission Control for the Data Bus (DBUS0)	511

24.3.1.3	Permission Control for On-chip DMA	512
24.3.1.4	Permission Control for PeriBus1	513
24.3.1.5	Permission Control for PeriBus2	515
24.3.1.6	Permission Control for Cache	516
24.3.1.7	Permission Control of Other Types of Internal Memory	516
24.3.2	External Memory Permission Control	517
24.3.2.1	Cache MMU	517
24.3.2.2	External Memory Permission Controls	517
24.3.3	Non-Aligned Access Permission Control	518
24.4	Base Address	519
24.5	Register Summary	519
24.6	Registers	521
<b>25</b>	<b>Digital Signature</b>	<b>549</b>
25.1	Overview	549
25.2	Features	549
25.3	Functional Description	549
25.3.1	Overview	549
25.3.2	Private Key Operands	549
25.3.3	Conventions	550
25.3.4	Software Storage of Private Key Data	550
25.3.5	DS Operation at the Hardware Level	551
25.3.6	DS Operation at the Software Level	552
25.4	Base Address	553
25.5	Memory Blocks	553
25.6	Register Summary	553
25.7	Registers	554
<b>26</b>	<b>HMAC Module</b>	<b>556</b>
26.1	Overview	556
26.2	Main Features	556
26.3	Functional Description	556
26.3.1	Upstream Mode	556
26.3.2	Downstream JTAG Enable Mode	557
26.3.3	Downstream Digital Signature Mode	557
26.3.4	HMAC eFuse Configuration	558
26.3.5	HMAC Process (Detailed)	558
26.4	HMAC Algorithm Details	560
26.4.1	Padding Bits	560
26.4.2	HMAC Algorithm Structure	561
26.5	Base Address	562
26.6	Register Summary	562
26.7	Registers	563
<b>27</b>	<b>On-Chip Sensor and Analog Signal Processing</b>	<b>569</b>
27.1	Overview	569

27.2	SAR ADCs	569
27.2.1	Overview	569
27.2.2	Features	570
27.2.3	Functional Description	570
27.2.3.1	Input Signals	571
27.2.3.2	ADC Conversion and Attenuation	572
27.2.4	RTC ADC Controllers	572
27.2.5	DIG ADC Controllers	573
27.2.5.1	Workflow of DIG ADC Controller	573
27.2.5.2	DMA	575
27.2.5.3	ADC Filter	575
27.2.5.4	Threshold Monitoring	576
27.2.6	SAR ADC2 Arbiter	576
27.3	DACs	577
27.3.1	Overview	577
27.3.2	Features	577
27.3.3	DAC Conversion	578
27.3.4	Cosine Wave Generator	578
27.3.5	DMA Support	579
27.4	Temperature Sensor	579
27.4.1	Overview	579
27.4.2	Features	579
27.4.3	Operation Sequence	580
27.4.4	Temperature Conversion	580
27.5	Interrupts	581
27.6	Base Address	581
27.7	Register Summary	581
27.7.1	SENSOR (RTC_PERI) Register Summary	582
27.7.2	SENSOR (DIG_PERI) Register Summary	582
27.8	Register	583
27.8.1	SENSOR (RTC_PERI) Registers	583
27.8.2	SENSOR (DIG_PERI) Registers	591
<b>28</b>	<b>ULP Coprocessor</b>	<b>603</b>
28.1	Overview	603
28.2	Features	603
28.3	Programming Workflow	605
28.4	ULP Coprocessor Workflow	605
28.5	ULP-FSM	607
28.5.1	Features	607
28.5.2	Instruction Set	607
28.5.2.1	ALU - Perform Arithmetic and Logic Operations	608
28.5.2.2	ST – Store Data in Memory	611
28.5.2.3	LD – Load Data from Memory	613
28.5.2.4	JUMP – Jump to an Absolute Address	614
28.5.2.5	JUMPR – Jump to a Relative Offset (Conditional upon R0)	614

28.5.2.6	JUMPS – Jump to a Relative Address (Conditional upon Stage Count Register)	615
28.5.2.7	HALT – End the Program	616
28.5.2.8	WAKE – Wake up the Chip	616
28.5.2.9	WAIT – Wait for a Number of Cycles	616
28.5.2.10	TSENS – Take Measurement with Temperature Sensor	617
28.5.2.11	ADC – Take Measurement with ADC	617
28.5.2.12	REG_RD – Read from Peripheral Register	618
28.5.2.13	REG_WR – Write to Peripheral Register	619
28.6	ULP-RISC-V	619
28.6.1	Features	619
28.6.2	Multiplier and Divider	619
28.6.3	ULP-RISC-V Interrupts	620
28.7	RTC I2C Controller	621
28.7.1	Connecting RTC I2C Signals	621
28.7.2	Configuring RTC I2C	621
28.7.3	Using RTC I2C	621
28.7.3.1	Instruction Format	621
28.7.3.2	I2C_RD - I2C Read Workflow	621
28.7.3.3	I2C_WR - I2C Write Workflow	622
28.7.3.4	Detecting Error Conditions	623
28.7.4	RTC I2C Interrupts	623
28.8	Base Address	624
28.8.1	ULP Coprocessor Base Address	624
28.8.2	RTC I2C Base Address	624
28.9	Register Summary	624
28.9.1	ULP (ALWAYS_ON) Register Summary	624
28.9.2	ULP (RTC_PERI) Register Summary	625
28.9.3	RTC I2C (RTC_PERI) Register Summary	625
28.9.4	RTC I2C (I2C) Register Summary	625
28.10	Registers	626
28.10.1	ULP (ALWAYS_ON) Registers	627
28.10.2	ULP (RTC_PERI) Registers	630
28.10.3	RTC I2C (RTC_PERI) Registers	633
28.10.4	RTC I2C (I2C) Registers	635
<b>29</b>	<b>Low-power Management</b>	<b>649</b>
29.1	Introduction	649
29.2	Features	649
29.3	Functional Description	649
29.3.1	Power Management Unit	650
29.3.2	Low-Power Clocks	652
29.3.3	Timers	653
29.3.4	Regulators	654
29.3.4.1	Digital System Voltage Regulator	654
29.3.4.2	Low-power Voltage Regulator	655
29.3.4.3	Flash Voltage Regulator	656

29.3.4.4	Brownout Detector	657
29.4	Power Modes Management	658
29.4.1	Power Domain	658
29.4.2	RTC States	658
29.4.3	Pre-defined Power Modes	660
29.4.4	Wakeup Source	660
29.5	RTC Boot	662
29.6	Base Address	663
29.7	Register Summary	663
29.8	Registers	665
<b>Revision History</b>		702



## List of Tables

1	Address Mapping . . . . .	25
2	Internal Memory Address Mapping . . . . .	25
3	External Memory Address Mapping . . . . .	27
4	Peripherals with DMA Support . . . . .	29
5	Module / Peripheral Address Mapping . . . . .	30
6	Addresses with Restricted Access . . . . .	32
7	Reset Source . . . . .	34
8	CPU_CLK Source . . . . .	36
9	CPU_CLK Selection . . . . .	36
10	Peripheral Clock Usage . . . . .	37
11	APB_CLK Source . . . . .	37
12	REF_TICK Source . . . . .	38
13	LEDC_PWM_CLK Source . . . . .	38
14	Default Configuration of Strapping Pins . . . . .	40
15	Boot Mode . . . . .	40
16	ROM Code Printing Control . . . . .	41
17	CPU Peripheral Interrupt Configuration/Status Registers and Peripheral Interrupt Sources . . . . .	43
18	CPU Interrupts . . . . .	46
19	Interrupt Matrix Base Address . . . . .	48
21	Pin Function Register for IO MUX Light-sleep Mode . . . . .	99
22	GPIO Matrix . . . . .	101
23	IO MUX Pad List . . . . .	105
24	RTC IO MUX Pin Summary . . . . .	106
25	GPIO, IO MUX, GPIOSD, Dedicated GPIO, and RTCIO Base Addresses . . . . .	107
31	ROM Controlling Bit . . . . .	151
32	SRAM Controlling Bit . . . . .	151
33	Peripheral Clock Gating and Reset Bits . . . . .	153
34	System Register Base Address . . . . .	155
36	Relationship Between Configuration Register, Block Size and Alignment . . . . .	176
37	Copy DMA and Crypto DMA Base Address . . . . .	180
40	UART0, UART1 and UHCI0 Base Address . . . . .	218
43	LED_PWM Base Address . . . . .	270
45	RMT Base Address . . . . .	283
47	Counter Mode. Positive Edge of Input Pulse Signal. Control Signal in Low State . . . . .	296
48	Counter Mode. Positive Edge of Input Pulse Signal. Control Signal in High State . . . . .	296
49	Counter Mode. Negative Edge of Input Pulse Signal. Control Signal in Low State . . . . .	296
50	Counter Mode. Negative Edge of Input Pulse Signal. Control Signal in High State . . . . .	296
51	PCNT Base Address . . . . .	299
53	64-bit Timers Base Address . . . . .	311
55	System Timer Base Address . . . . .	330
57	Parameters in BLOCK0 . . . . .	339
58	Key Purpose Values . . . . .	342
59	Parameters in BLOCK1-10 . . . . .	342
61	Configuration of eFuse-Programming Timing Parameters . . . . .	347

62	Configuration of VDDQ Timing Parameters . . . . .	348
63	Configuration of eFuse-Reading Parameters . . . . .	348
64	eFuse Controller Base Address . . . . .	349
66	I2C Controller Base Address . . . . .	389
68	Data Frames and Remote Frames in SFF and EFF . . . . .	416
69	Error Frame . . . . .	417
70	Overload Frame . . . . .	418
71	Interframe Space . . . . .	419
72	Segments of a Nominal Bit Time . . . . .	421
73	Bit Information of TWAI_CLOCK_DIVIDER_REG; TWAI Address 0x18 . . . . .	425
74	Bit Information of TWAI_BUS_TIMING_1_REG; TWAI Address 0x1c . . . . .	426
75	Buffer Layout for Standard Frame Format and Extended Frame Format . . . . .	428
76	TX/RX Frame Information (SFF/EFF) TWAI Address 0x40 . . . . .	429
77	TX/RX Identifier 1 (SFF); TWAI Address 0x44 . . . . .	429
78	TX/RX Identifier 2 (SFF); TWAI Address 0x48 . . . . .	430
79	TX/RX Identifier 1 (EFF); TWAI Address 0x44 . . . . .	430
80	TX/RX Identifier 2 (EFF); TWAI Address 0x48 . . . . .	430
81	TX/RX Identifier 3 (EFF); TWAI Address 0x4c . . . . .	430
82	TX/RX Identifier 4 (EFF); TWAI Address 0x50 . . . . .	430
83	Bit Information of TWAI_ERR_CODE_CAP_REG; TWAI Address 0x30 . . . . .	435
84	Bit Information of Bits SEG.4 - SEG.0 . . . . .	435
85	Bit Information of TWAI_ARB LOST CAP_REG; TWAI Address 0x2c . . . . .	436
86	TWAI Base Address . . . . .	437
88	AES Accelerator Working Mode . . . . .	452
89	Operation Type under Typical AES Working Mode . . . . .	452
90	Working Status under Typical AES Working Mode . . . . .	452
91	Text Endianness Types for Typical AES . . . . .	453
92	Key Endianness Types for AES-128 Encryption and Decryption . . . . .	455
93	Key Endianness Types for AES-192 Encryption and Decryption . . . . .	455
94	Key Endianness Types for AES-256 Encryption and Decryption . . . . .	456
95	Operation Type under DMA-AES Working Mode . . . . .	458
96	Working Status under DMA-AES Working mode . . . . .	458
97	TEXT-PADDING . . . . .	459
98	Text Endianness for DMA-AES . . . . .	459
99	AES Accelerator Base Address . . . . .	464
100	AES Accelerator Memory Blocks . . . . .	464
102	SHA Accelerator Working Mode . . . . .	472
103	SHA Hash Algorithm . . . . .	472
107	The Storage and Length of Message digest from Different Algorithms . . . . .	479
108	SHA Accelerator Base Address . . . . .	480
110	Acceleration Performace . . . . .	490
111	RSA Accelerator Base Address . . . . .	491
112	RSA Accelerator Memory Blocks . . . . .	491
114	Random Number Generator Base Address . . . . .	497
116	Key . . . . .	499
117	Mapping Between Offsets and Registers . . . . .	501

118	Manual Encryption Block Base Address . . . . .	503
120	Offset Address Range of Each SRAM Block . . . . .	508
121	Permission Control for IBUS to Access SRAM . . . . .	509
122	Permission Control for IBUS to Access RTC FAST Memory . . . . .	510
123	Permission Control for DBUS0 to Access SRAM . . . . .	511
124	Permission Control for DBUS0 to Access RTC FAST Memory . . . . .	512
125	Permission Control for On-chip DMA to Access SRAM . . . . .	513
126	Peripherals and FIFO Address . . . . .	514
127	Permission Control for PeriBus1 . . . . .	514
128	Permission Control for PeriBus2 to Access RTC SLOW Memory . . . . .	515
129	Configuration of Register PMS_PRO_CACHE_1_REG . . . . .	516
130	MMU Entries . . . . .	517
131	Non-Aligned Access to Peripherals . . . . .	519
132	Permission Control Base Address . . . . .	519
134	Digital Signature Base Address . . . . .	553
137	HMAC Function and Configuration Value . . . . .	558
138	HMAC Base Address . . . . .	562
140	SAR ADC Controllers . . . . .	570
141	SAR ADC Input Signals . . . . .	571
142	Fields of Pattern Table Register . . . . .	574
143	DMA Data Format (Type I) . . . . .	575
144	DMA Data Format (Type II) . . . . .	575
145	Temperature Offset . . . . .	581
146	On-Chip Sensor, SAR ADCs, and DACs Base Addresses . . . . .	581
149	Comparison of the Two Coprocessors . . . . .	604
150	ALU Operations Among Registers . . . . .	609
151	ALU Operations with Immediate Value . . . . .	610
152	ALU Operations with Stage Count Register . . . . .	610
153	Data Storage Type - Automatic Storage Mode . . . . .	612
154	Data Storage - Manual Storage Mode . . . . .	613
155	Input Signals Measured Using the ADC Instruction . . . . .	617
156	Instruction Efficiency . . . . .	620
157	ULP-RISC-V Interrupt List . . . . .	620
158	ULP Coprocessor Base Address . . . . .	624
159	RTC I2C Base Address . . . . .	624
164	Low-Power Clocks . . . . .	653
165	The Triggering Conditions for the RTC Timer . . . . .	653
166	Brown-out Detector Configuration . . . . .	657
167	RTC Statues Transition . . . . .	659
168	Predefined Power Modes . . . . .	660
169	Wakeup Source . . . . .	661
170	Low-power Management Base Address . . . . .	663

## List of Figures

1-1	System Structure and Address Mapping . . . . .	24
1-2	Cache Structure . . . . .	28
2-1	System Reset . . . . .	33
2-2	System Clock . . . . .	35
4-1	Interrupt Matrix Structure . . . . .	42
5-1	IO MUX, RTC IO MUX and GPIO Matrix Overview . . . . .	90
5-2	GPIO Input Synchronized on Clock Rising Edge or on Falling Edge . . . . .	92
5-3	Filter Timing Diagram of GPIO Input Signals . . . . .	92
5-4	Dedicated GPIO Diagram . . . . .	96
7-1	Modules with DMA and Supported Data Transfers . . . . .	171
7-2	DMA Engine Architecture . . . . .	172
7-3	Structure of a Linked List . . . . .	172
7-4	Relationship among Linked Lists . . . . .	174
7-5	EDMA Receiving Data Frames in Internal . . . . .	175
7-6	Copy DMA Engine Architecture . . . . .	176
7-7	Data Transfer in UDMA Mode . . . . .	177
7-8	SPI DMA . . . . .	177
8-1	UART Structure . . . . .	208
8-2	UART Controllers Sharing RAM . . . . .	209
8-3	UART Controllers Division . . . . .	210
8-4	The Timing Diagram of Weak UART Signals Along Falling Edges . . . . .	211
8-5	Structure of UART Data Frame . . . . .	211
8-6	AT_CMD Character Structure . . . . .	212
8-7	Driver Control Diagram in RS485 Mode . . . . .	213
8-8	The Timing Diagram of Encoding and Decoding in SIR mode . . . . .	214
8-9	IrDA Encoding and Decoding Diagram . . . . .	214
8-10	Hardware Flow Control Diagram . . . . .	215
8-11	Connection between Hardware Flow Control Signals . . . . .	215
9-1	LED_PWM Architecture . . . . .	267
9-2	LED_PWM generator Diagram . . . . .	268
9-3	LED_PWM Divider . . . . .	268
9-4	LED_PWM Output Signal Diagram . . . . .	269
9-5	Output Signal Diagram of Fading Duty Cycle . . . . .	270
10-1	RMT Architecture . . . . .	280
10-2	Format of Pulse Code in RAM . . . . .	281
11-1	PCNT Block Diagram . . . . .	294
11-2	PCNT Unit Architecture . . . . .	295
11-3	Channel 0 Up Counting Diagram . . . . .	297
11-4	Channel 0 Down Counting Diagram . . . . .	298
11-5	Two Channels Up Counting Diagram . . . . .	298
12-1	Timer Units within Groups . . . . .	307
13-1	Super Watchdog Controller Structure . . . . .	325
14-1	XTAL32K Watchdog . . . . .	326
15-1	System Timer Structure . . . . .	328

16-1	Shift Register Circuit . . . . .	344
16-2	eFuse-Programming Timing Diagram . . . . .	347
16-3	Timing Diagram for Reading eFuse . . . . .	349
17-1	I2C Master Architecture . . . . .	376
17-2	I2C Slave Architecture . . . . .	376
17-3	Structure of I2C Command Register . . . . .	377
17-4	I2C Timing Diagram . . . . .	379
17-5	An I2C Master Writing to an I2C Slave with a 7-bit Address . . . . .	381
17-6	A Master Writing to a Slave with a 10-bit Address . . . . .	382
17-7	An I2C Master Writing Address M in the RAM to an I2C Slave with a 7-bit Address . . . . .	383
17-8	An I2C Master Writing to an I2C Slave with a 7-bit Address in Multiple Sequences . . . . .	383
17-9	An I2C Master Reading an I2C Slave with a 7-bit Address . . . . .	384
17-10	An I2C Master Reading an I2C Slave with a 10-bit Address . . . . .	385
17-11	An I2C Master Reading N Bytes of Data from addrM of an I2C Slave with a 7-bit Address . . . . .	386
17-12	An I2C Master Reading an I2C Slave with a 7-bit Address in Segments . . . . .	387
18-1	The bit fields of Data Frames and Remote Frames . . . . .	415
18-2	Various Fields of an Error Frame . . . . .	417
18-3	The Bit Fields of an Overload Frame . . . . .	418
18-4	The Fields within an Interframe Space . . . . .	419
18-5	Layout of a Bit . . . . .	421
18-6	TWAI Overview Diagram . . . . .	423
18-7	Acceptance Filter . . . . .	431
18-8	Single Filter Mode . . . . .	432
18-9	Dual Filter Mode . . . . .	433
18-10	Error State Transition . . . . .	434
18-11	Positions of Arbitration Lost Bits . . . . .	436
19-12	GCM Encryption Process . . . . .	462
22-1	Noise Source . . . . .	496
23-1	Architecture of the External Memory Encryption and Decryption Module . . . . .	498
25-1	Preparations and DS Operation . . . . .	550
26-1	HMAC SHA-256 Padding Diagram . . . . .	560
26-2	HMAC Structure Schematic Diagram . . . . .	561
27-1	SAR ADC Overview . . . . .	569
27-2	SAR ADC Function Overview . . . . .	571
27-3	RTC SAR ADC Outline . . . . .	572
27-4	Diagram of DIG ADC Controllers . . . . .	574
27-5	Diagram of DAC Function . . . . .	578
27-6	Workflow of CW Generator . . . . .	578
27-7	Structure of Temperature Sensor . . . . .	580
28-1	ULP Coprocessor Overview . . . . .	603
28-2	ULP Coprocessor Diagram . . . . .	604
28-3	Programing Workflow . . . . .	605
28-4	Sample of a ULP Operation Sequence . . . . .	606
28-5	Control of ULP Program Execution . . . . .	607
28-6	ULP-FSM Instruction Format . . . . .	608
28-7	Instruction Type — ALU for Operations Among Registers . . . . .	608

28-8	Instruction Type — ALU for Operations with Immediate Value . . . . .	609
28-9	Instruction Type — ALU for Operations with Stage Count Register . . . . .	610
28-10	Instruction Type - ST . . . . .	611
28-11	Instruction Type - Offset in Automatic Storage Mode (ST-OFFSET) . . . . .	611
28-12	Instruction Type - Data Storage in Automatic Storage Mode (ST-AUTO-DATA) . . . . .	611
28-13	Instruction Type - Data Storage in Manual Storage Mode . . . . .	612
28-14	Instruction Type - LD . . . . .	613
28-15	Instruction Type- JUMP . . . . .	614
28-16	Instruction Type - JUMPR . . . . .	614
28-17	Instruction Type - JUMPS . . . . .	615
28-18	Instruction Type- HALT . . . . .	616
28-19	Instruction Type - WAKE . . . . .	616
28-20	Instruction Type - WAIT . . . . .	616
28-21	Instruction Type - TSENS . . . . .	617
28-22	Instruction Type - ADC . . . . .	617
28-23	Instruction Type - REG_RD . . . . .	618
28-24	Instruction Type - REG_WR . . . . .	619
28-25	I2C Read Operation . . . . .	622
28-26	I2C Write Operation . . . . .	623
29-1	Low-power Management Schematics . . . . .	650
29-2	Power Management Unit Workflow . . . . .	651
29-3	Low-Power Clocks for RTC Power Domains . . . . .	652
29-4	Low-Power Clocks for RTC Power Domains . . . . .	652
29-5	Low-Power Clocks for RTC Fast Memory and Slow Memory . . . . .	652
29-6	Digital System Regulator . . . . .	655
29-7	Low-power voltage regulator . . . . .	655
29-8	Flash voltage regulator . . . . .	656
29-9	Brown-out detector . . . . .	657
29-10	RTC States . . . . .	659
29-11	ESP32-S2 Boot Flow . . . . .	663

# 1. System and Memory

## 1.1 Overview

The ESP32-S2 is a single-core system with one Harvard Architecture Xtensa® LX7 CPU. All internal memory, external memory, and peripherals are located on the CPU buses.

## 1.2 Features

- **Address Space**
  - 4 GB (32 bits wide) address space in total accessed from the data bus and instruction bus
  - 464 KB internal memory address space accessed from the instruction bus
  - 400 KB internal memory address space accessed from the data bus
  - 1.77 MB peripheral address space
  - 7.5 MB external memory virtual address space accessed from the instruction bus
  - 14.5 MB external memory virtual address space accessed from the data bus
  - 320 KB internal DMA address space
  - 10.5 MB external DMA address space
- **Internal Memory**
  - 128 KB Internal ROM
  - 320 KB Internal SRAM
  - 8 KB RTC FAST Memory
  - 8 KB RTC SLOW Memory
- **External Memory**
  - Supports up to 1 GB external SPI flash
  - Supports up to 1 GB external SPI RAM
- **DMA**
  - 9 DMA-supported modules / peripherals

Figure 1-1 illustrates the system structure and address mapping.

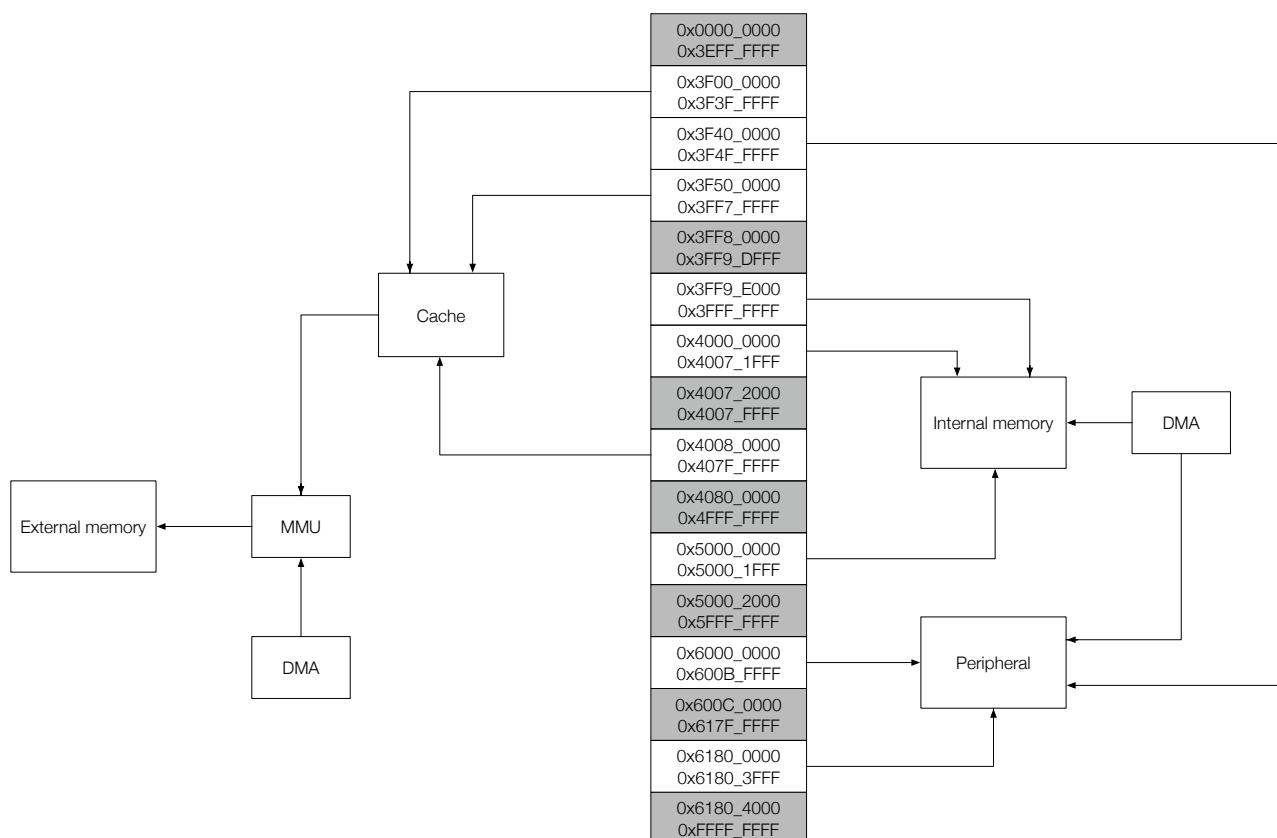


Figure 1-1. System Structure and Address Mapping

**Note:**

- The memory space with gray background is not available to users.
- The range of addresses available in the address space may be larger or smaller than the actual available memory of a particular type.

## 1.3 Functional Description

### 1.3.1 Address Mapping

The Harvart Architecture Xtensa® LX7 CPU can address 4 GB (32 bits wide) memory space.

Addresses below 0x4000\_0000 are serviced using the data bus. Addresses in the range 0x4000\_0000 ~ 0x4FFF\_FFFF are serviced using the instruction bus. Addresses over and including 0x5000\_0000 are shared by both data and instruction bus.

Both data bus and instruction bus are little-endian. The CPU can access data via the data bus in a byte-, half-word-, or word-aligned manner. The CPU can also access data via the instruction bus, but only in a word-aligned manner; non-word-aligned access will cause a CPU exception.

The CPU can:

- directly access the internal memory via both data bus and instruction bus;
- access the external memory which is mapped into the address space via cache;



- access modules / peripherals via data bus.

Table 1 lists the address ranges on the data bus and instruction bus and their corresponding target memory.

Some internal and external memory can be accessed via both data bus and instruction bus. In such cases, the same memory is available to the CPU at two address ranges.

**Table 1: Address Mapping**

Bus Type	Boundary Address		Size	Target
	Low Address	High Address		
	0x0000_0000	0x3EFF_FFFF		Reserved
Data bus	0x3F00_0000	0x3F3F_FFFF	4 MB	External memory
	0x3F40_0000	0x3F4F_FFFF	1 MB	Peripherals
	0x3F50_0000	0x3FF7_FFFF	10.5 MB	External memory
	0x3FF8_0000	0x3FF9_DFFF		Reserved
Data bus	0x3FF9_E000	0x3FFF_FFFF	392 KB	Internal memory
Instruction bus	0x4000_0000	0x4007_1FFF	456 KB	Internal memory
	0x4007_2000	0x4007_FFFF		Reserved
Instruction bus	0x4008_0000	0x407F_FFFF	7.5 MB	External memory
	0x4080_0000	0x4FFF_FFFF		Reserved
Data / Instruction bus	0x5000_0000	0x5000_1FFF	8 KB	Internal memory
	0x5000_2000	0x5FFF_FFFF		Reserved
Data / Instruction bus	0x6000_0000	0x600B_FFFF	768 KB	Peripherals
	0x600C_0000	0x617F_FFFF		Reserved
Data / Instruction bus	0x6180_0000	0x6180_3FFF	16 KB	Peripherals
	0x6180_4000	0xFFFF_FFFF		Reserved

### 1.3.2 Internal Memory

The internal memory consists of four segments: Internal ROM (128 KB), Internal SRAM (320 KB), RTC FAST Memory (8 KB), and RTC SLOW Memory (8 KB).

The Internal ROM is broken down into two parts: Internal ROM 0 (64 KB) and Internal ROM 1 (64 KB).

The Internal SRAM is broken down into two parts: Internal SRAM 0 (32 KB) and Internal SRAM 1 (288 KB).

RTC FAST Memory and RTC SLOW Memory are both implemented as SRAM.

Table 2 lists all types of internal memory and their address ranges on the data bus and instruction bus.

**Table 2: Internal Memory Address Mapping**

Bus Type	Boundary Address		Size	Target	Permission Control
	Low Address	High Address			
Data bus	0x3FF9_E000	0x3FF9_FFFF	8 KB	RTC FAST Memory	YES
	0x3FFA_0000	0x3FFA_FFFF	64 KB	Internal ROM 1	NO
	0x3FFB_0000	0x3FFB_7FFF	32 KB	Internal SRAM 0	YES
	0x3FFB_8000	0x3FFF_FFFF	288 KB	Internal SRAM 1	YES

Bus Type	Boundary Address		Size	Target	Permission Control
	Low Address	High Address			
Instruction bus	0x4000_0000	0x4000_FFFF	64 KB	Internal ROM 0	NO
	0x4001_0000	0x4001_FFFF	64 KB	Internal ROM 1	NO
	0x4002_0000	0x4002_7FFF	32 KB	Internal SRAM 0	YES
	0x4002_8000	0x4006_FFFF	288 KB	Internal SRAM 1	YES
	0x4007_0000	0x4007_1FFF	8 KB	RTC FAST Memory	YES
Bus Type	Boundary Address		Size	Target	Permission Control
	Low Address	High Address			
Data / Instruction bus	0x5000_0000	0x5000_1FFF	8 KB	RTC SLOW Memory	YES

**Note:**

"YES" in the "Permission Control" column indicates that a permission is required for memory access. Permission Control registers can be used to limit Instruction or Data bus access to individual regions of these memory types.

### 1.3.2.1 Internal ROM 0

Internal ROM 0 is a 64-KB, read-only memory space, addressed by the CPU on the instruction bus via range(s) described in Table 2.

### 1.3.2.2 Internal ROM 1

Internal ROM 1 is a 64-KB, read-only memory space, addressed by the CPU on the data or instruction bus via range(s) described in Table 2.

The two address ranges access Internal ROM 1 in the same order, so, for example, addresses 0x3FFA\_0000 and 0x4001\_0000 access the same word, 0x3FFA\_0004 and 0x4001\_0004 access the same word, 0x3FFA\_0008 and 0x4001\_0008 access the same word, etc.

### 1.3.2.3 Internal SRAM 0

Internal SRAM 0 is a 32-KB, read-and-write memory space, addressed by the CPU on the data or instruction bus, in the same order, via range(s) described in Table 2.

Hardware can be configured to use 8 KB, 16 KB, 24 KB, or the entire 32 KB space in this memory to cache external memory. The space used as cache cannot be accessed by the CPU, while the remaining space can still be accessed by the CPU.

### 1.3.2.4 Internal SRAM 1

Internal SRAM 1 is a 288-KB, read-and-write memory space, addressed by the CPU on the data or instruction bus, in the same order, via range(s) described in Table 2.

Internal SRAM 1 comprises eighteen 16-KB (sub)memory blocks. One block can be used as Trace Memory, in which case this block's address range cannot be accessed by the CPU.

### 1.3.2.5 RTC FAST Memory

RTC FAST Memory is an 8-KB, read-and-write SRAM, addressed by the CPU on the data or instruction bus, in the same order, via range(s) described in Table 2.

### 1.3.2.6 RTC SLOW Memory

RTC SLOW Memory is an 8-KB, read-and-write SRAM, addressed by the CPU via range(s) shared by the data bus and the instruction bus, as described in Table 2.

RTC SLOW Memory can also be used as a peripheral addressable to the CPU via either 0x3F42\_1000 ~ 0x3F42\_2FFF or 0x6002\_1000 ~ 0x6002\_2FFF on the data bus.

## 1.3.3 External Memory

ESP32-S2 supports multiple QSPI/OSPI flash and RAM chips. It also supports hardware encryption/decryption based on XTS-AES to protect user programs and data in the flash and external RAM.

### 1.3.3.1 External Memory Address Mapping

The CPU accesses the external flash and RAM via the cache. According to the MMU settings, the cache maps the CPU's address to the external physical memory address. Due to this address mapping, the ESP32-S2 can address up to 1 GB external flash and 1 GB external RAM.

Using the cache, ESP32-S2 can support the following address space mappings at the same time.

- Up to 7.5 MB instruction bus address space can be mapped into the external flash or RAM as individual 64 KB blocks, via the instruction cache (ICache). Byte (8-bit), half-word (16-bit) and word (32-bit) reads are supported.
- Up to 4 MB read-only data bus address space can be mapped into the external flash or RAM as individual 64 KB blocks, via ICache. Byte (8-bit), half-word (16-bit) and word (32-bit) reads are supported.
- Up to 10.5 MB data bus address space can be mapped into the external RAM as individual 64 KB blocks, via DCache. Byte (8-bit), half-word (16-bit) or word (32-bit) reads and writes are supported. Blocks from this 10.5 MB space can also be mapped into the external flash or RAM, for read operations only.

Table 3 lists the mapping between the cache and the corresponding address ranges on the data bus and instruction bus.

**Table 3: External Memory Address Mapping**

Bus Type	Boundary Address		Size	Target	Permission Control
	Low Address	High Address			
Data bus	0x3F00_0000	0x3F3F_FFFF	4 MB	ICache	YES
Data bus	0x3F50_0000	0x3FF7_FFFF	10.5 MB	DCache	YES
Instruction bus	0x4008_0000	0x407F_FFFF	7.5 MB	ICache	YES

**Note:**

"YES" in the "Permission Control" column indicates that a permission is required for memory access. Permission Control registers can be used to limit Instruction or Data bus access to individual regions of these memory types.

### 1.3.3.2 Cache

As shown in Figure 1-2, the caches on ESP32-S2 are separated and allow prompt response upon simultaneous requests from the data bus and instruction bus. Some internal memory space can be used as cache (see Section 1.3.2.3). When a cache miss occurs, the cache controller will initiate a request to the external memory. When ICache and DCache simultaneously initiate a request, the arbiter determines which gets the access to the external memory first. The cache size of ICache and DCache can be configured to 8 KB and 16 KB, respectively, while their block size can be configured to 16 bytes and 32 bytes, respectively.

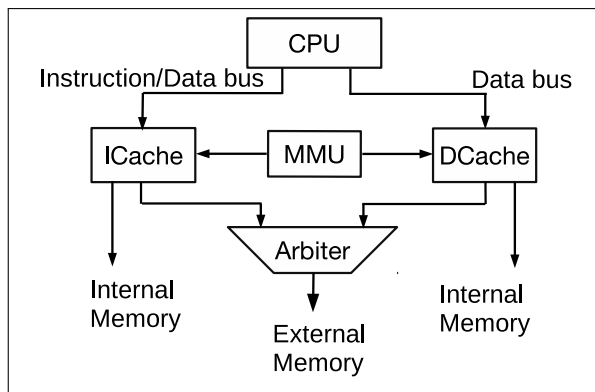


Figure 1-2. Cache Structure

### 1.3.3.3 Cache Operations

ESP32-S2 caches support the following operations:

1. **Invalidate:** The cache clears the valid bit of a tag. The CPU needs to access the external memory in order to read/write the data. There are two types of invalidation operations: manual invalidation and automatic invalidation. Manual invalidation performs only on data in the specified area in the cache, while automatic invalidation performs on all data in the cache. Both ICache and DCache have this function.
2. **Clean:** The cache clears the dirty bit of the tag and retains the valid bit. The CPU can then read/write the data directly from the cache. Only DCache has this function.
3. **Write-back:** The cache clears the dirty block flag of the tag and retains the valid bit. It also forces the data in the corresponding address to be written back to the external memory. The CPU can then read/write the data directly from the cache. Only DCache has this function.
4. **Preload:** To preload a cache is to load instructions and data into the cache in advance. The minimum unit of a preloading is one block. There are two types of preloading: manual preloading and automatic preloading. Manual preloading means that the hardware prefetches a piece of continuous data according to the virtual address specified by the software. Automatic preloading means the hardware prefetches a piece of continuous data according to the current hit / miss address (depending on configuration).
5. **Lock / Unlock:** There are two types of lock: prelock and manual lock. When prelock is enabled, the cache locks the data in the specified area when filling the missing data to cache memory. When manual lock is enabled, the cache checks the data that has been filled into the cache memory and locks the data that falls in the specified area. The data in the locked area is always stored in the cache and will not be replaced. But when all the ways within the cache are locked, the cache will replace data, as if the ways were not locked. Unlocking is the reverse of locking. The manual invalidation, clean, and write-back operations are only available after unlocking.

### 1.3.4 DMA

With DMA, the ESP32-S2 can perform data transfers between:

- modules / peripherals and internal memory;
- different types of internal memory;
- modules / peripherals and external memory;
- internal and external memory.

DMA uses the same addressing as the data bus to read and write Internal SRAM 0 and Internal SRAM 1. Specifically, DMA uses address range 0x3FFB\_0000 ~ 0x3FFB\_7FFF to access Internal SRAM 0, and 0x3FFB\_8000 ~ 0x3FFF\_FFFF to access Internal SRAM 1. Note that DMA cannot access the internal memory occupied by the cache.

In addition, DMA addresses the external RAM from 0x3F50\_0000 ~ 0x3FF7\_FFFF, the same used by the CPU to access DCache. When DCache and DMA access the external memory simultaneously, data consistency is required.

Nine modules / peripherals on the ESP32-S2 support DMA, as shown in Table 4. With DMA, some of them can only access internal memory, some can access both internal and external memory.

**Table 4: Peripherals with DMA Support**

UART0	UART1
SPI2	SPI3
I2S0	
ADC Controller	
Copy DMA	
AES Accelerator	SHA Accelerator

### 1.3.5 Modules / Peripherals

The CPU can access modules / peripherals via address range 0x3F40\_0000 ~ 0x3F4F\_FFFF on the data bus, or via 0x6000\_0000 ~ 0x600B\_FFFF and 0x6180\_0000 ~ 0x6180\_3FFF shared by the data bus and instruction bus.

#### 1.3.5.1 Naming Conventions for Peripheral Buses

There are two peripheral buses defined as follows:

- **PeriBus1**: Which refers to the address range 0x3F40\_0000 ~ 0x3F4F\_FFFF on the bus. 0x3F40\_0000 is the base address.
- **PeriBus2**: Which refers to the address ranges 0x6000\_0000 ~ 0x600B\_FFFF and 0x6180\_0000 ~ 0x6180\_3FFF on the bus. 0x6000\_0000 is the base address.

All references to “[PeriBus1](#)” and “[PeriBus2](#)” in this document indicate the corresponding address range(s).

### 1.3.5.2 Differences Between PeriBus1 and PeriBus2

The CPU can access modules / peripherals more efficiently through PeriBus1 than through PeriBus2. However, PeriBus1 features speculative reads, which means it cannot guarantee that each read is valid. Therefore, the CPU has to use PeriBus2 to access some special registers, for example, FIFO registers.

In addition, PeriBus1 will upset the order of r/w operations on the bus to improve performance, which may cause programs that have strict requirements on the r/w order to crash. In such cases, please add volatile before the program statement, or use PeriBus2 instead.

### 1.3.5.3 Module / Peripheral Address Mapping

Table 5 lists all the modules / peripherals and their respective address ranges. Note that addresses in column “Boundary Address” are offsets relative to the base address, instead of absolute addresses. The absolute addresses are the addition of bus base address and the corresponding offsets.

**Table 5: Module / Peripheral Address Mapping**

Target	Boundary Address		Size	Notes
	Low Address	High Address		
UART0	0x0000_0000	0x0000_0FFF	4 KB	1, 2, 3
Reserved	0x0000_1000	0x0000_1FFF		
SPI1	0x0000_2000	0x0000_2FFF	4 KB	1, 2
SPI0	0x0000_3000	0x0000_3FFF	4 KB	1, 2
GPIO	0x0000_4000	0x0000_4FFF	4 KB	1, 2
Reserved	0x0000_5000	0x0000_6FFF		
TIMER	0x0000_7000	0x0000_7FFF	4 KB	1, 2
RTC	0x0000_8000	0x0000_8FFF	4 KB	1, 2
IO MUX	0x0000_9000	0x0000_9FFF	4 KB	1, 2
Reserved	0x0000_A000	0x0000_EFFF		
I2S0	0x0000_F000	0x0000_FFFF	4 KB	1, 2, 3
UART1	0x0001_0000	0x0001_0FFF	4 KB	1, 2, 3
Reserved	0x0001_1000	0x0001_2FFF		
I2C0	0x0001_3000	0x0001_3FFF	4 KB	1, 2, 3
UHCIO	0x0001_4000	0x0001_4FFF	4 KB	1, 2
Reserved	0x0001_5000	0x0001_5FFF		
RMT	0x0001_6000	0x0001_6FFF	4 KB	1, 2, 3
PCNT	0x0001_7000	0x0001_7FFF	4 KB	1, 2
Reserved	0x0001_8000	0x0001_8FFF		
LED PWM Controller	0x0001_9000	0x0001_9FFF	4 KB	1, 2
eFuse Controller	0x0001_A000	0x0001_AFFF	4 KB	1, 2
Reserved	0x0001_B000	0x0001_EFFF		
Timer Group 0	0x0001_F000	0x0001_FFFF	4 KB	1, 2
Timer Group 1	0x0002_0000	0x0002_0FFF	4 KB	1, 2
RTC SLOW Memory	0x0002_1000	0x0002_2FFF	8 KB	1, 2, 3
System Timer	0x0002_3000	0x0002_3FFF	4 KB	1, 2
SPI2	0x0002_4000	0x0002_4FFF	4 KB	1, 2

Target	Boundary Address		Size	Notes
	Low Address	High Address		
SPI3	0x0002_5000	0x0002_5FFF	4 KB	1, 2
APB Controller	0x0002_6000	0x0002_6FFF	4 KB	1, 2
I2C1	0x0002_7000	0x0002_7FFF	4 KB	1, 2, 3
Reserved	0x0002_8000	0x0002_AFFF		
TWAI Controller	0x0002_B000	0x0002_BFFF	4 KB	1, 2
Reserved	0x0002_C000	0x0003_8FFF		
USB OTG	0x0003_9000	0x0003_9FFF	4 KB	1, 2, 3, 4
AES Accelerator	0x0003_A000	0x0003_AFFF	4 KB	1, 2
SHA Accelerator	0x0003_B000	0x0003_BFFF	4 KB	1, 2
RSA Accelerator	0x0003_C000	0x0003_CFFF	4 KB	1, 2
Digital Signature	0x0003_D000	0x0003_DFFF	4 KB	1, 2
HMAC	0x0003_E000	0x0003_EFFF	4 KB	1, 2
Crypto DMA	0x0003_F000	0x0003_FFFF	4 KB	1, 2
Reserved	0x0004_4000	0x000C_DFFF		
ADC Controller	0x0004_0000	0x0004_0FFF	4 KB	1, 2
Reserved	0x0004_1000	0x0007_FFFF		
USB OTG	0x0008_0000	0x000B_FFFF	256 KB	1, 2, 3, 4
System Registers	0x000C_0000	0x000C_0FFF	4 KB	1
Sensitive Register	0x000C_1000	0x000C_1FFF	4 KB	1
Interrupt Matrix	0x000C_2000	0x000C_2FFF	4 KB	1
Copy DMA	0x000C_3000	0x000C_3FFF	4 KB	1
Reserved	0x000C_4000	0x000C_EFFF		
Dedicated GPIO	0x000C_F000	0x000C_FFFF	4 KB	1
Reserved	0x000D_1000	0x000F_FFFF		
Configure Cache	0x0180_0000	0x0180_3FFF	16 KB	2

**Note:**

1. This module / peripheral can be accessed from [PeriBus1](#).
2. This module / peripheral can be accessed from [PeriBus2](#).
3. Some special addresses in this module / peripheral are not accessible from [PeriBus1](#) (see Section 1.3.5.4).
4. The address space in this module / peripheral is not continuous.

### 1.3.5.4 Addresses with Restricted Access from [PeriBus1](#)

As mentioned in Section 1.3.5.2, [PeriBus1](#) features speculative reads, which means it is forbidden to read FIFO registers. Table 6 below lists the address (range) with restricted access from [PeriBus1](#).

There are four reserved user-defined registers that can be configured as needed to add more addresses with restricted access.

**Table 6: Addresses with Restricted Access**

Peripherals	Addresses with Restricted Access
UART0	0x3F40_0000
UART1	0x3F41_0000
I2S0	0x3F40_F004
RMT	0x3F41_6000 ~ 0x3F41_600F
I2C0	0x3F41_301C
I2C1	0x3F42_701C
USB OTG	0x3F48_0020, 0x3F48_1000 ~ 0x3F49_0FFF



## 2. Reset and Clock

### 2.1 Reset

#### 2.1.1 Overview

ESP32-S2 provides four types of reset that occur at different levels, namely CPU Reset, Core Reset, System Reset, and Chip Reset.

All reset types mentioned above (except Chip Reset) maintain the data stored in internal memory. Figure 2-1 shows the scopes of affected subsystems when different types of reset occur.

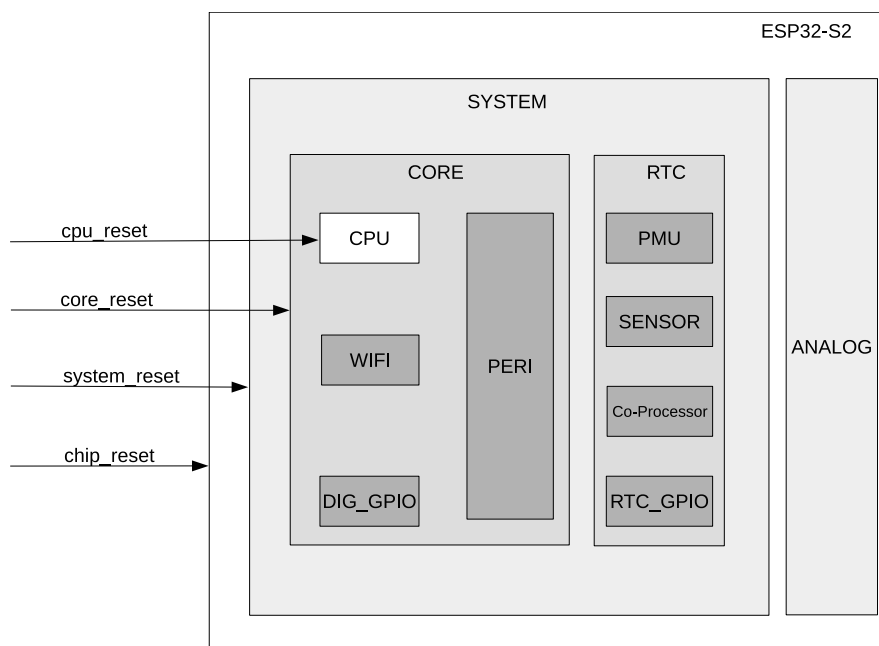


Figure 2-1. System Reset

- CPU Reset: Only resets CPU core. Once such reset is released, programs will be executed from CPU reset vector.
- Core Reset: Resets the whole digital system except RTC, including CPU, peripherals, Wi-Fi, and digital GPIOs.
- System Reset: Resets the whole digital system, including RTC.
- Chip Reset: Resets the whole chip.

#### 2.1.2 Reset Source

CPU will be reset immediately when any of the reset above occurs. Users can get reset source codes by reading register [RTC\\_CNTL\\_RESET\\_CAUSE\\_PROCPU](#) after the reset is released.

Table 7 lists different reset sources and the types of reset they trigger.

**Table 7: Reset Source**

Code	Source	Reset Type	Comments
0x01	Chip reset	Chip Reset	See the note below
0x0F	Brown-out system reset	System Reset	Triggered by brown-out detector
0x10	RWDT system reset	System Reset	See Chapter <a href="#">13 Watchdog Timers</a>
0x13	GLITCH reset	System Reset	-
0x03	Software system reset	Core Reset	Triggered by configuring <code>RTC_CNTL_SW_SYS_RST</code>
0x05	Deep-sleep reset	Core Reset	See Chapter <a href="#">29 Low-power Management</a>
0x07	MWDT0 global reset	Core Reset	See Chapter <a href="#">13 Watchdog Timers</a>
0x08	MWDT1 global reset	Core Reset	See Chapter <a href="#">13 Watchdog Timers</a>
0x09	RWDT core reset	Core Reset	See Chapter <a href="#">13 Watchdog Timers</a>
0x0B	MWDT0 CPU reset	CPU Reset	See Chapter <a href="#">13 Watchdog Timers</a>
0x0C	Software CPU reset	CPU Reset	Triggered by configuring <code>RTC_CNTL_SW_PROCPU_RST</code>
0x0D	RWDT CPU reset	CPU Reset	See Chapter <a href="#">13 Watchdog Timers</a>
0x11	MWDT1 CPU reset	CPU Reset	See Chapter <a href="#">13 Watchdog Timers</a>

Note:

- Chip Reset can be triggered by the following three sources:
  - Triggered by chip power-on;
  - Triggered by brown-out detector;
  - Triggered by Super Watchdog (SWD).
- Once brown-out status is detected, the detector will trigger System Reset or Chip Reset, depending on register configuration. For more information, please see Chapter [29 Low-power Management](#).

## 2.2 Clock

### 2.2.1 Overview

ESP32-S2 provides multiple clock sources, which allow CPU, peripherals and RTC to work at different frequencies, thus providing more flexibility in meeting the requirements of various application scenarios. Figure [2-2](#) shows the system clock structure.

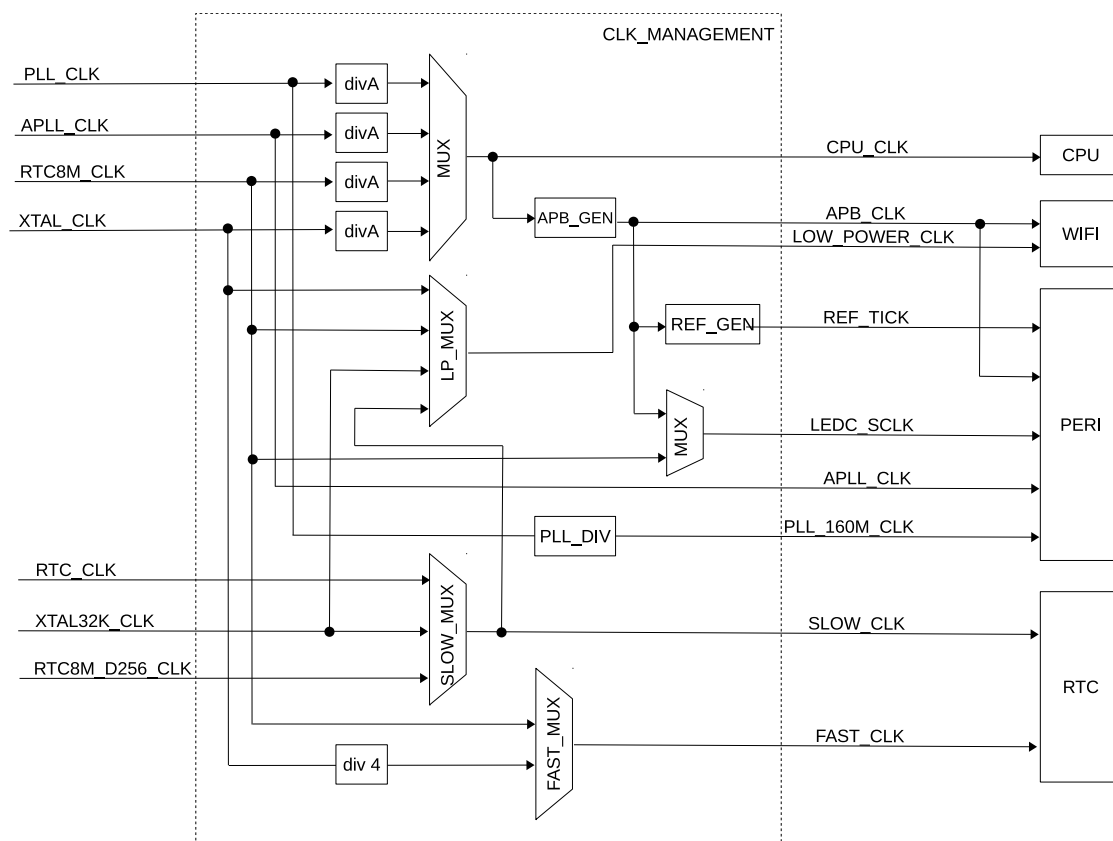


Figure 2-2. System Clock

### 2.2.2 Clock Source

ESP32-S2 uses external crystal, internal PLL, or internal oscillator working as clock sources to generate different kinds of clocks, which can be classified in three types depending on their clock speed.

- High speed clock for devices working at a higher frequency, such as CPU and digital peripherals
  - *PLL\_CLK* (320 MHz or 480 MHz): internal PLL clock
  - *XTAL\_CLK* (40 MHz): external crystal clock
- Slow speed clock for low-power devices, such as power management unit and low-power peripherals
  - *XTAL32K\_CLK* (32 kHz): external crystal clock
  - *RTC8M\_CLK* (8 MHz by default): internal oscillator with adjustable frequency
  - *RTC8M\_D256\_CLK* (31.250 kHz by default): internal clock derived from *RTC8M\_CLK* divided by 256
  - *RTC\_CLK* (90 kHz by default): internal oscillator with adjustable frequency
- Audio clock for audio-related devices
  - *APLL\_CLK* (16 MHz ~ 128 MHz): internal Audio PLL clock

### 2.2.3 CPU Clock

As Figure 2-2 shows, *CPU\_CLK* is the master clock for CPU and it can be as high as 240 MHz when CPU works in high performance mode. Alternatively, CPU can run at lower frequencies, such as at 2 MHz, to lower power consumption.

Users can set PLL\_CLK, APLL\_CLK, RTC8M\_CLK or XTAL\_CLK as CPU\_CLK clock source by configuring register [SYSTEM\\_SOC\\_CLK\\_SEL](#), see Table 8 and Table 9.

**Table 8: CPU\_CLK Source**

<a href="#">SYSTEM_SOC_CLK_SEL</a> Value	Clock Source
0	XTAL_CLK
1	PLL_CLK
2	RTC8M_CLK
3	APLL_CLK

**Table 9: CPU\_CLK Selection**

Clock Source	SEL_0*	SEL_1*	SEL_2*	CPU Clock Frequency
XTAL_CLK	0	-	-	$CPU\_CLK = XTAL\_CLK / (SYSTEM\_PRE\_DIV\_CNT + 1)$ <a href="#">SYSTEM_PRE_DIV_CNT</a> ranges from 0 ~ 1023. Default is 1
PLL_CLK (480 MHz)	1	1	0	$CPU\_CLK = PLL\_CLK / 6$ CPU_CLK frequency is 80 MHz
PLL_CLK (480 MHz)	1	1	1	$CPU\_CLK = PLL\_CLK / 3$ CPU_CLK frequency is 160 MHz
PLL_CLK (480 MHz)	1	1	2	$CPU\_CLK = PLL\_CLK / 2$ CPU_CLK frequency is 240 MHz
PLL_CLK (320 MHz)	1	0	0	$CPU\_CLK = PLL\_CLK / 4$ CPU_CLK frequency is 80 MHz
PLL_CLK (320 MHz)	1	0	1	$CPU\_CLK = PLL\_CLK / 2$ CPU_CLK frequency is 160 MHz
RTC8M_CLK	2	-	-	$CPU\_CLK = RTC8M\_CLK / (SYSTEM\_PRE\_DIV\_CNT + 1)$ <a href="#">SYSTEM_PRE_DIV_CNT</a> ranges from 0 ~ 1023. Default is 1
APLL_CLK	3	0	0	$CPU\_CLK = APLL\_CLK / 4$
APLL_CLK	3	0	1	$CPU\_CLK = APLL\_CLK / 2$

\*SEL\_0: The value of register [SYSTEM\\_SOC\\_CLK\\_SEL](#).

\*SEL\_1: The value of register [SYSTEM\\_PLL\\_FREQ\\_SEL](#).

\*SEL\_2: The value of register [SYSTEM\\_CPUPERIOD\\_SEL](#).

Note:

- When users select XTAL\_CLK as CPU clock source and adjust the divider value by configuring register [SYSTEM\\_PRE\\_DIV\\_CNT](#), the rules below should be followed.
  - If current divider value is 2 ( $SYSTEM\_PRE\_DIV\_CNT = 1$ ) and the target is  $x$  ( $x \neq 1$ ), users should set the divider first to 1 ( $SYSTEM\_PRE\_DIV\_CNT = 0$ ) and then to  $x$  ( $SYSTEM\_PRE\_DIV\_CNT = x - 1$ ).
  - If current divider value is  $x$  ( $SYSTEM\_PRE\_DIV\_CNT = x - 1$ ) and the target is 2, users should set the divider first to 1 ( $SYSTEM\_PRE\_DIV\_CNT = 0$ ) and then to 2 ( $SYSTEM\_PRE\_DIV\_CNT = 1$ ).
  - For other target divider value  $x$ , users can adjust the register directly ( $SYSTEM\_PRE\_DIV\_CNT = x - 1$ ).

### 2.2.4 Peripheral Clock

Peripheral clocks include APB\_CLK, REF\_TICK, LEDC\_PWM\_CLK, APLL\_CLK and PLL\_160M\_CLK. Table 10 shows which clock can be used by which peripheral.

**Table 10: Peripheral Clock Usage**

Peripheral	APB_CLK	REF_TICK	LEDC_PWM_CLK	APLL_CLK	PLL_160M_CLK
TIMG	Y	Y			
I <sup>2</sup> S	Y			Y	Y
UHCI	Y				
UART	Y	Y			
RMT	Y	Y			
LED_PWM	Y	Y	Y		
I <sup>2</sup> C	Y	Y			
SPI	Y				
PCNT	Y				
eFuse Controller	Y				
SARADC/DAC	Y			Y	
USB	Y				
CRYPTO					Y
TWAI Controller	Y				
System Timer	Y				

#### 2.2.4.1 APB\_CLK Source

APB\_CLK is determined by the clock source of CPU\_CLK as shown in Table 11.

**Table 11: APB\_CLK Source**

CPU_CLK Source	APB_CLK
PLL_CLK	80 MHz
APLL_CLK	CPU_CLK / 2
XTAL_CLK	CPU_CLK
RTC8M_CLK	CPU_CLK

#### 2.2.4.2 REF\_TICK Source

REF\_TICK is derived from XTAL\_CLK or RTC8M\_CLK via a divider. When PLL\_CLK, APLL\_CLK or XTAL\_CLK is set as CPU clock source, REF\_TICK will be divided from XTAL\_CLK. When RTC8M\_CLK is set as CPU clock source, REF\_TICK will be divided from RTC8M\_CLK. In such way, REF\_TICK frequency remains unchanged when APB\_CLK changes its clock source. Table 12 shows the configuration of these clock divider registers.

**Table 12: REF\_TICK Source**

CPU_CLK Source	Clock Divider Register
PLL_CLK   XTAL_CLK   APLL_CLK	APB_CTRL_XTAL_TICK_NUM
RTC8M_CLK	APB_CTRL_CK8M_TICK_NUM

Normally, one REF\_TICK cycle lasts for 1  $\mu$ s, so APB\_CTRL\_XTAL\_TICK\_NUM should be configured to 39 (default), and APB\_CTRL\_CK8M\_TICK\_NUM to 7 (default).

### 2.2.4.3 LEDC\_PWM\_CLK Source

LEDC\_PWM\_CLK clock source is selected by configuring register [LEDC\\_APB\\_CLK\\_SEL](#), as shown in Table 13.

**Table 13: LEDC\_PWM\_CLK Source**

<a href="#">LEDC_APB_CLK_SEL</a> Value	LEDC_PWM_CLK Source
0 (Default)	-
1	APB_CLK
2	RTC8M_CLK
3	XTAL_CLK

### 2.2.4.4 APLL\_SCLK Source

APLL\_CLK is sourced from PLL\_CLK, and its output frequency is configured using APLL configuration registers. See Section 2.2.7 for more information.

### 2.2.4.5 PLL\_160M\_CLK Source

PLL\_160M\_CLK is divided from PLL\_CLK according to current PLL frequency.

### 2.2.4.6 Clock Source Considerations

Peripherals that need to work with other clocks, such as RMT and I<sup>2</sup>C, generally operate using PLL\_CLK frequency as a reference. When this frequency changes, peripherals should update their clock configuration to operate at the same frequency after the change. Peripherals accessing REF\_TICK can continue operating normally without changing their clock configuration when switching clock sources. Please see Table 10.

LED module uses RTC8M\_CLK as clock source when APB\_CLK is disabled. In other words, when the system is in low-power mode, most peripherals will be halted (APB\_CLK is turned off), but LED can work normally via RTC8M\_CLK.

## 2.2.5 Wi-Fi Clock

Wi-Fi can work only when APB\_CLK uses PLL\_CLK as its clock source. Suspending PLL\_CLK requires that Wi-Fi has entered low-power mode first.

LOW\_POWER\_CLK uses XTAL32K\_CLK, XTAL\_CLK, RTC8M\_CLK or SLOW\_CLK (the low clock selected by RTC) as its clock source for Wi-Fi in low-power mode.

### 2.2.6 RTC Clock

The clock sources for SLOW\_CLK and FAST\_CLK are low-frequency clocks. RTC module can operate when most other clocks are stopped.

SLOW\_CLK derived from RTC\_CLK, XTAL32K\_CLK or RTC8M\_D256\_CLK is used to clock Power Management module.

FAST\_CLK is used to clock On-chip Sensor module. It can be sourced from a divided XTAL\_CLK or from RTC8M\_CLK.

### 2.2.7 Audio PLL Clock

The operation of audio and other time-critical data-transfer applications requires highly-configurable, low-jitter and accurate clock sources. The clock sources derived from system clocks that serve digital peripherals may carry jitter and, therefore, are not suitable for a high-precision clock frequency setting.

Providing an integrated precision clock source can minimize system cost. To this end, ESP32-S2 integrates an audio PLL to clock I<sup>2</sup>S module.

Audio PLL formula is as follows:

$$f_{\text{out}} = \frac{f_{\text{xtal}}(\text{sdm2} + \frac{\text{sdm1}}{2^8} + \frac{\text{sdm0}}{2^{16}} + 4)}{2(\text{odiv} + 2)}$$

Parameters are defined below:

- $f_{\text{xtal}}$ : the frequency of crystal oscillator, usually 40 MHz;
- sdm0: the value is 0 ~ 255;
- sdm1: the value is 0 ~ 255;
- sdm2: the value is 0 ~ 63;
- odiv: the value is 0 ~ 31;
- The operating frequency range of the numerator is 350 MHz ~ 500 MHz.

$$350MHz < f_{\text{xtal}}(\text{sdm2} + \frac{\text{sdm1}}{2^8} + \frac{\text{sdm0}}{2^{16}} + 4) < 500MHz$$

Audio PLL can be manually enabled or disabled via registers [RTC\\_CNTL\\_PLLA\\_FORCE\\_PU](#) and [RTC\\_CNTL\\_PLLA\\_FORCE\\_PD](#), respectively. Disabling it takes priority over enabling it. When [RTC\\_CNTL\\_PLLA\\_FORCE\\_PU](#) and [RTC\\_CNTL\\_PLLA\\_FORCE\\_PD](#) are 0, PLL follows the state of the system. When the system enters sleep mode, PLL will be disabled automatically; when the system wakes up, PLL will be enabled automatically.

## 3. Chip Boot Control

### 3.1 Overview

ESP32-S2 has three strapping pins:

- GPIO0
- GPIO45
- GPIO46

Software can read the values of the three strapping pins from register [GPIO\\_STRAPPING](#). During chip reset triggered by power-on, brown-out or by analog super watchdog (see Chapter [2 Reset and Clock](#)), hardware samples and stores the voltage level of strapping pins as strapping bit of “0” or “1” in latches, and hold these bits until the chip is powered down or shut down.

By default, GPIO0, GPIO45 and GPIO46 are connected to internal pull-up/pull-down during chip reset. Consequently, if the three GPIOs are unconnected or their connected external circuits are high-impedance, the internal weak pull-up/pull-down will determine the default input level of the strapping pins (see Table [14](#)).

**Table 14: Default Configuration of Strapping Pins**

Pin	GPIO0	GPIO45	GPIO46
Default	Pull-up	Pull-down	Pull-down

To change the default configuration of strapping pins, users can apply external pull-down/pull-up resistors, or use host MCU GPIOs to control the voltage level of these pins when powering on ESP32-S2. After the reset is released, the strapping pins work as normal-function pins.

### 3.2 Boot Mode

GPIO0 and GPIO46 control the boot mode after reset.

**Table 15: Boot Mode**

Pin	SPI Boot	Download Boot
GPIO0	1	0
GPIO46	x	0

Table [15](#) shows the strapping pin values and the associated boot modes. “x” means that this value is ignored. Currently only the two boot modes shown are supported. The strapping combination of GPIO0 = 0 and GPIO46 = 1 is not supported and will trigger unexpected behavior.

In SPI boot mode, the CPU boots the system by reading the program stored in SPI flash.

In download boot mode, users can download code to SRAM or Flash using UART0, UART1, QPI or USB interface. It is also possible to load a program into SRAM and execute it in this mode.

The following eFuses control boot mode behavior:

- [EFUSE\\_DIS\\_FORCE\\_DOWNLOAD](#). If this eFuse is 0 (default), software can switch the chip from SPI boot mode to download boot mode by setting register [RTC\\_CNTL\\_FORCE\\_DOWNLOAD\\_BOOT](#) and triggering



a CPU reset. If this eFuse is 1, this register is disabled.

- [EFUSE\\_DIS\\_DOWNLOAD\\_MODE](#). If this eFuse is 1, download boot mode is disabled.
- [EFUSE\\_ENABLE\\_SECURITY\\_DOWNLOAD](#). If this eFuse is 1, download boot mode only allows reading, writing and erasing plaintext flash and does not support any SRAM or register operations. This eFuse is ignored if download boot mode is disabled.

### 3.3 ROM Code Printing to UART

GPIO46 controls ROM code printing of information during the early boot process. This GPIO is used together with the [UART\\_PRINT\\_CONTROL](#) eFuse.

**Table 16: ROM Code Printing Control**

UART_PRINT_CONTROL	GPIO46	ROM Code Printing
0	-	ROM code will always print information to UART during boot. GPIO46 is not used.
1	0	Print is enabled during boot
	1	Print is disabled
2	0	Print is disabled
	1	Print is enabled during boot
3	-	Print is always disabled during boot. GPIO46 is not used.

ROM code will print to pin U0TXD (default) or to pin DAC\_1, depending on the eFuse bit [UART\\_PRINT\\_CHANNEL](#) (0: UART0; 1: DAC\_1).

### 3.4 VDD\_SPI Voltage

GPIO45 is used to select the VDD\_SPI power supply voltage at reset:

- GPIO45 = 0, VDD\_SPI pin is powered directly from VDD3P3\_RTC\_IO via resistor  $R_{SPI}$ . Typically this voltage is 3.3 V.
- GPIO45 = 1, VDD\_SPI pin is powered from internal 1.8 V LDO.

This functionality can be overridden by setting eFuse bit [VDD\\_SPI\\_FORCE](#) to 1, in which case the [VDD\\_SPI\\_TIEH](#) eFuse value determines the VDD\_SPI voltage:

- [VDD\\_SPI\\_FORCE](#) = 1 and [VDD\\_SPI\\_TIEH](#) = 0, VDD\_SPI connects to 1.8 V LDO.
- [VDD\\_SPI\\_FORCE](#) = 1 and [VDD\\_SPI\\_TIEH](#) = 1, VDD\_SPI connects to VDD3P3\_RTC\_IO.

## 4. Interrupt Matrix

### 4.1 Overview

The interrupt matrix embedded in ESP32-S2 independently allocates peripheral interrupt sources to the CPU peripheral interrupts, so as to timely inform the CPU to process the interrupts once the interrupt signals are generated. This flexible function is applicable to a variety of application scenarios.

### 4.2 Features

- Accept 95 peripheral interrupt sources as input
- Generate 26 peripheral interrupts to the CPU as output
- Disable CPU non-maskable interrupt (NMI) sources
- Query current interrupt status of peripheral interrupt sources

The structure of the interrupt matrix is shown in Figure 4-1.

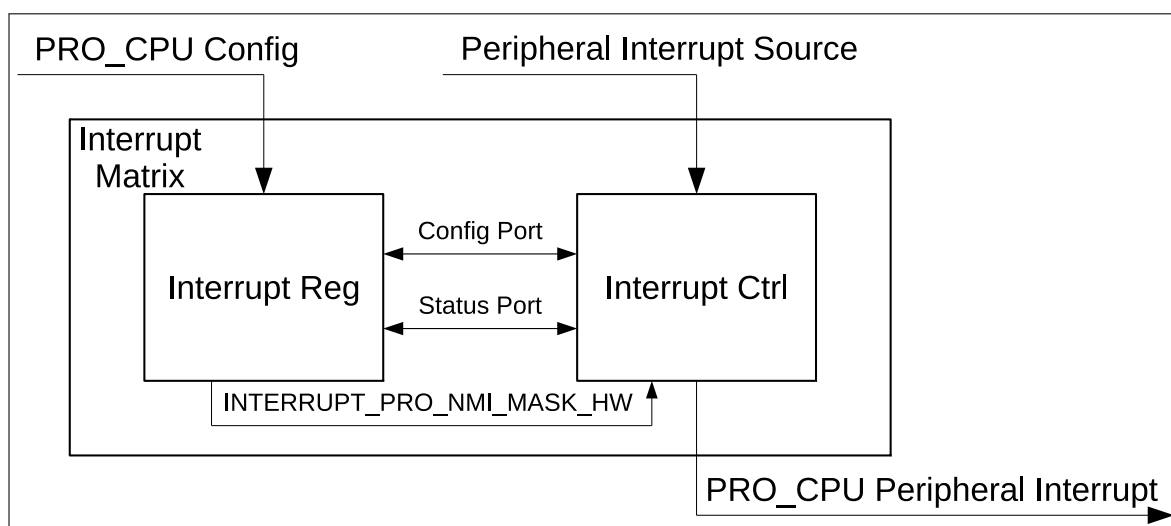


Figure 4-1. Interrupt Matrix Structure

### 4.3 Functional Description

#### 4.3.1 Peripheral Interrupt Sources

ESP32-S2 has 95 peripheral interrupt sources in total, all of which can be allocated to the CPU. For the peripheral interrupt sources and their configuration/status registers, please refer to Table 17.

Table 17: CPU Peripheral Interrupt Configuration/Status Registers and Peripheral Interrupt Sources

No.	Source	Configuration Register	Bit	Status Register Name
0	MAC_INTR	<a href="#">INTERRUPT_PRO_MAC_INTR_MAP_REG</a>	0	<a href="#">INTERRUPT_PRO_INTR_STATUS_REG_0_REG</a>
1	MAC_NMI	<a href="#">INTERRUPT_PRO_MAC_NMI_MAP_REG</a>	1	
2	PWR_INTR	<a href="#">INTERRUPT_PRO_PWR_INTR_MAP_REG</a>	2	
3	BB_INT	<a href="#">INTERRUPT_PRO_BB_INT_MAP_REG</a>	3	
4	BT_MAC_INT	<a href="#">INTERRUPT_PRO_BT_MAC_INT_MAP_REG</a>	4	
5	BT_BB_INT	<a href="#">INTERRUPT_PRO_BT_BB_INT_MAP_REG</a>	5	
6	BT_BB_NMI	<a href="#">INTERRUPT_PRO_BT_BB_NMI_MAP_REG</a>	6	
7	RWBT_IRQ	<a href="#">INTERRUPT_PRO_RWBT_IRQ_MAP_REG</a>	7	
8	RWBLE_IRQ	<a href="#">INTERRUPT_PRO_RWBLE_IRQ_MAP_REG</a>	8	
9	RWBT_NMI	<a href="#">INTERRUPT_PRO_RWBT_NMI_MAP_REG</a>	9	
10	RWBLE_NMI	<a href="#">INTERRUPT_PRO_RWBLE_NMI_MAP_REG</a>	10	
11	SLC0_INTR	<a href="#">INTERRUPT_PRO_SLC0_INTR_MAP_REG</a>	11	
12	SLC1_INTR	<a href="#">INTERRUPT_PRO_SLC1_INTR_MAP_REG</a>	12	
13	UHCIO_INTR	<a href="#">INTERRUPT_PRO_UHCIO_INTR_MAP_REG</a>	13	
14	UHCI1_INTR	<a href="#">INTERRUPT_PRO_UHCI1_INTR_MAP_REG</a>	14	
15	TG_T0_LEVEL_INT	<a href="#">INTERRUPT_PRO_TG_T0_LEVEL_INT_MAP_REG</a>	15	
16	TG_T1_LEVEL_INT	<a href="#">INTERRUPT_PRO_TG_T1_LEVEL_INT_MAP_REG</a>	16	
17	TG_WDT_LEVEL_INT	<a href="#">INTERRUPT_PRO_TG_WDT_LEVEL_INT_MAP_REG</a>	17	
18	TG_LACT_LEVEL_INT	<a href="#">INTERRUPT_PRO_TG_LACT_LEVEL_INT_MAP_REG</a>	18	
19	TG1_T0_LEVEL_INT	<a href="#">INTERRUPT_PRO_TG1_T0_LEVEL_INT_MAP_REG</a>	19	
20	TG1_T1_LEVEL_INT	<a href="#">INTERRUPT_PRO_TG1_T1_LEVEL_INT_MAP_REG</a>	20	
21	TG1_WDT_LEVEL_INT	<a href="#">INTERRUPT_PRO_TG1_WDT_LEVEL_INT_MAP_REG</a>	21	
22	TG1_LACT_LEVEL_INT	<a href="#">INTERRUPT_PRO_TG1_LACT_LEVEL_INT_MAP_REG</a>	22	
23	GPIO_INTERRUPT_PRO	<a href="#">INTERRUPT_PRO_GPIO_INTERRUPT_PRO_MAP_REG</a>	23	
24	GPIO_INTERRUPT_PRO_NMI	<a href="#">INTERRUPT_PRO_GPIO_INTERRUPT_PRO_NMI_MAP_REG</a>	24	
25	GPIO_INTERRUPT_APP	<a href="#">INTERRUPT_PRO_GPIO_INTERRUPT_APP_MAP_REG</a>	25	
26	GPIO_INTERRUPT_APP_NMI	<a href="#">INTERRUPT_PRO_GPIO_INTERRUPT_APP_NMI_MAP_REG</a>	26	
27	DEDICATED_GPIO_IN_INTR	<a href="#">INTERRUPT_PRO_DEDICATED_GPIO_IN_INTR_MAP_REG</a>	27	
28	CPU_INTR_FROM_CPU_0	<a href="#">INTERRUPT_PRO_CPU_INTR_FROM_CPU_0_MAP_REG</a>	28	
29	CPU_INTR_FROM_CPU_1	<a href="#">INTERRUPT_PRO_CPU_INTR_FROM_CPU_1_MAP_REG</a>	29	
30	CPU_INTR_FROM_CPU_2	<a href="#">INTERRUPT_PRO_CPU_INTR_FROM_CPU_2_MAP_REG</a>	30	
31	CPU_INTR_FROM_CPU_3	<a href="#">INTERRUPT_PRO_CPU_INTR_FROM_CPU_3_MAP_REG</a>	31	
32	SPI_INTR_1	<a href="#">INTERRUPT_PRO_SPI_INTR_1_MAP_REG</a>	0	<a href="#">INTERRUPT_PRO_INTR_STATUS_REG_1_REG</a>
33	SPI_INTR_2	<a href="#">INTERRUPT_PRO_SPI_INTR_2_MAP_REG</a>	1	
34	SPI_INTR_3	<a href="#">INTERRUPT_PRO_SPI_INTR_3_MAP_REG</a>	2	

No.	Source	Configuration Register	Bit	Status Register Name
35	I2S0_INT	INTERRUPT_PRO_I2S0_INT_MAP_REG	3	INTERRUPT_PRO_INTR_STATUS_REG_1_REG
36	I2S1_INT	INTERRUPT_PRO_I2S1_INT_MAP_REG	4	
37	UART_INTR	INTERRUPT_PRO_UART_INTR_MAP_REG	5	
38	UART1_INTR	INTERRUPT_PRO_UART1_INTR_MAP_REG	6	
39	UART2_INTR	INTERRUPT_PRO_UART2_INTR_MAP_REG	7	
40	SDIO_HOST_INTERRUPT	INTERRUPT_PRO_SDIO_HOST_INTERRUPT_MAP_REG	8	
41	PWM0_INTR	INTERRUPT_PRO_PWM0_INTR_MAP_REG	9	
42	PWM1_INTR	INTERRUPT_PRO_PWM1_INTR_MAP_REG	10	
43	PWM2_INTR	INTERRUPT_PRO_PWM2_INTR_MAP_REG	11	
44	PWM3_INTR	INTERRUPT_PRO_PWM3_INTR_MAP_REG	12	
45	LEDC_INT	INTERRUPT_PRO_LEDC_INT_MAP_REG	13	
46	EFUSE_INT	INTERRUPT_PRO_EFUSE_INT_MAP_REG	14	
47	CAN_INT	INTERRUPT_PRO_CAN_INT_MAP_REG	15	
48	USB_INTR	INTERRUPT_PRO_USB_INTR_MAP_REG	16	
49	RTC_CORE_INTR	INTERRUPT_PRO_RTC_CORE_INTR_MAP_REG	17	
50	RMT_INTR	INTERRUPT_PRO_RMT_INTR_MAP_REG	18	
51	PCNT_INTR	INTERRUPT_PRO_PCNT_INTR_MAP_REG	19	
52	I2C_EXT0_INTR	INTERRUPT_PRO_I2C_EXT0_INTR_MAP_REG	20	
53	I2C_EXT1_INTR	INTERRUPT_PRO_I2C_EXT1_INTR_MAP_REG	21	
54	RSA_INTR	INTERRUPT_PRO_RSA_INTR_MAP_REG	22	
55	SHA_INTR	INTERRUPT_PRO_SHA_INTR_MAP_REG	23	
56	AES_INTR	INTERRUPT_PRO_AES_INTR_MAP_REG	24	
57	SPI2_DMA_INT	INTERRUPT_PRO_SPI2_DMA_INT_MAP_REG	25	
58	SPI3_DMA_INT	INTERRUPT_PRO_SPI3_DMA_INT_MAP_REG	26	
59	WDG_INT	INTERRUPT_PRO_WDG_INT_MAP_REG	27	
60	TIMER_INT	INTERRUPT_PRO_TIMER_INT1_MAP_REG	28	
61	TIMER_INT2	INTERRUPT_PRO_TIMER_INT2_MAP_REG	29	
62	TG_T0_EDGE_INT	INTERRUPT_PRO_TG_T0_EDGE_INT_MAP_REG	30	
63	TG_T1_EDGE_INT	INTERRUPT_PRO_TG_T1_EDGE_INT_MAP_REG	31	
64	TG_WDT_EDGE_INT	INTERRUPT_PRO_TG_WDT_EDGE_INT_MAP_REG	0	INTERRUPT_PRO_INTR_STATUS_REG_2_REG
65	TG_LACT_EDGE_INT	INTERRUPT_PRO_TG_LACT_EDGE_INT_MAP_REG	1	
66	TG1_T0_EDGE_INT	INTERRUPT_PRO_TG1_T0_EDGE_INT_MAP_REG	2	
67	TG1_T1_EDGE_INT	INTERRUPT_PRO_TG1_T1_EDGE_INT_MAP_REG	3	
68	TG1_WDT_EDGE_INT	INTERRUPT_PRO_TG1_WDT_EDGE_INT_MAP_REG	4	
69	TG1_LACT_EDGE_INT	INTERRUPT_PRO_TG1_LACT_EDGE_INT_MAP_REG	5	
70	CACHE_IA_INT	INTERRUPT_PRO_CACHE_IA_INT_MAP_REG	6	
71	SYSTIMER_TARGET0_INT	INTERRUPT_PRO_SYSTIMER_TARGET0_INT_MAP_REG	7	

No.	Source	Configuration Register	Bit	Status Register Name
72	SYSTIMER_TARGET1_INT	<a href="#">INTERRUPT_PRO_SYSTIMER_TARGET1_INT_MAP_REG</a>	8	<a href="#">INTERRUPT_PRO_INTR_STATUS_REG_2_REG</a>
73	SYSTIMER_TARGET2_INT	<a href="#">INTERRUPT_PRO_SYSTIMER_TARGET2_INT_MAP_REG</a>	9	
74	ASSIST_DEBUG_INTR	<a href="#">INTERRUPT_PRO_ASSIST_DEBUG_INTR_MAP_REG</a>	10	
75	PMS_PRO_IRAM0_ILG_INTR	<a href="#">INTERRUPT_PRO_PMS_PRO_IRAM0_ILG_INTR_MAP_REG</a>	11	
76	PMS_PRO_DRAM0_ILG_INTR	<a href="#">INTERRUPT_PRO_PMS_PRO_DRAM0_ILG_INTR_MAP_REG</a>	12	
77	PMS_PRO_DPORT_ILG_INTR	<a href="#">INTERRUPT_PRO_PMS_PRO_DPORT_ILG_INTR_MAP_REG</a>	13	
78	PMS_PRO_AHB_ILG_INTR	<a href="#">INTERRUPT_PRO_PMS_PRO_AHB_ILG_INTR_MAP_REG</a>	14	
79	PMS_PRO_CACHE_ILG_INTR	<a href="#">INTERRUPT_PRO_PMS_PRO_CACHE_ILG_INTR_MAP_REG</a>	15	
80	PMS_DMA_APB_I_ILG_INTR	<a href="#">INTERRUPT_PRO_PMS_DMA_APB_I_ILG_INTR_MAP_REG</a>	16	
81	PMS_DMA_RX_I_ILG_INTR	<a href="#">INTERRUPT_PRO_PMS_DMA_RX_I_ILG_INTR_MAP_REG</a>	17	
82	PMS_DMA_TX_I_ILG_INTR	<a href="#">INTERRUPT_PRO_PMS_DMA_TX_I_ILG_INTR_MAP_REG</a>	18	
83	SPI_MEM_REJECT_INTR	<a href="#">INTERRUPT_PRO_SPI_MEM_REJECT_INTR_MAP_REG</a>	19	
84	DMA_COPY_INTR	<a href="#">INTERRUPT_PRO_DMA_COPY_INTR_MAP_REG</a>	20	
85	SPI4_DMA_INT	<a href="#">INTERRUPT_PRO_SPI4_DMA_INT_MAP_REG</a>	21	
86	SPI_INTR_4	<a href="#">INTERRUPT_PRO_SPI_INTR_4_MAP_REG</a>	22	
87	DCACHE_PRELOAD_INT	<a href="#">INTERRUPT_PRO_DCACHE_PRELOAD_INT_MAP_REG</a>	23	
88	ICACHE_PRELOAD_INT	<a href="#">INTERRUPT_PRO_ICACHE_PRELOAD_INT_MAP_REG</a>	24	
89	APB_ADC_INT	<a href="#">INTERRUPT_PRO_APB_ADC_INT_MAP_REG</a>	25	
90	CRYPTO_DMA_INT	<a href="#">INTERRUPT_PRO_CRYPTODMA_INT_MAP_REG</a>	26	
91	CPU_PERI_ERROR_INT	<a href="#">INTERRUPT_PRO_CPU_PERI_ERROR_INT_MAP_REG</a>	27	
92	APB_PERI_ERROR_INT	<a href="#">INTERRUPT_PRO_APB_PERI_ERROR_INT_MAP_REG</a>	28	
93	DCACHE_SYNC_INT	<a href="#">INTERRUPT_PRO_DCACHE_SYNC_INT_MAP_REG</a>	29	
94	ICACHE_SYNC_INT	<a href="#">INTERRUPT_PRO_ICACHE_SYNC_INT_MAP_REG</a>	30	

### 4.3.2 CPU Interrupts

The CPU has 32 interrupts, including 26 peripheral interrupts and six internal interrupts. Table 18 lists all the interrupts.

- Peripheral Interrupts:
  - Level-triggered interrupts: triggered by high level signal. The interrupt sources should hold the level till the CPU handles the interrupts.
  - Edge-triggered interrupts: triggered on rising edge. The CPU responds to this kind of interrupts immediately.
  - NMI interrupt: once triggered, the NMI interrupt can not be masked by software using the CPU registers.
- Internal Interrupts:
  - Timer interrupts: triggered by internal timers and are used to generate periodic interrupts.
  - Software interrupts: triggered when software writes to special registers.
  - Profiling interrupt: triggered for performance monitoring and analysis.

ESP32-S2 supports the above-mentioned 32 interrupts at six levels as shown in the table below. A higher level corresponds to a higher priority. NMI has the highest interrupt priority and once triggered, the CPU must handle such interrupt.

**Table 18: CPU Interrupts**

No.	Category	Type	Priority Level
0	Peripheral	Level-triggered	1
1	Peripheral	Level-triggered	1
2	Peripheral	Level-triggered	1
3	Peripheral	Level-triggered	1
4	Peripheral	Level-triggered	1
5	Peripheral	Level-triggered	1
6	Internal	Timer.0	1
7	Internal	Software	1
8	Peripheral	Level-triggered	1
9	Peripheral	Level-triggered	1
10	Peripheral	Edge-triggered	1
11	Internal	Profiling	3
12	Peripheral	Level-triggered	1
13	Peripheral	Level-triggered	1
14	Peripheral	NMI	NMI
15	Internal	Timer.1	3
16	Internal	Timer.2	5
17	Peripheral	Level-triggered	1
18	Peripheral	Level-triggered	1
19	Peripheral	Level-triggered	2
20	Peripheral	Level-triggered	2

No.	Category	Type	Priority Level
21	Peripheral	Level-triggered	2
22	Peripheral	Edge-triggered	3
23	Peripheral	Level-triggered	3
24	Peripheral	Level-triggered	4
25	Peripheral	Level-triggered	4
26	Peripheral	Level-triggered	5
27	Peripheral	Level-triggered	3
28	Peripheral	Edge-triggered	4
29	Internal	Software	3
30	Peripheral	Edge-triggered	4
31	Peripheral	Level-triggered	5

### 4.3.3 Allocate Peripheral Interrupt Source to CPU Interrupt

In this section, the following terms are used to describe the operation of the interrupt matrix.

- Source\_X: stands for a particular peripheral interrupt source, wherein, X means the number of this interrupt source in Table 17.
- INTERRUPT\_PRO\_X\_MAP\_REG: stands for a peripheral interrupt configuration register, corresponding to the peripheral interrupt source Source\_X. In Table 17, the registers listed in column "Configuration Register" correspond to the peripheral interrupt sources listed in column "Source". For example, the configuration register for source MAC\_INTR is [INTERRUPT\\_PRO\\_MAC\\_INTR\\_MAP\\_REG](#).
- Interrupt\_P: stands for the CPU peripheral interrupt numbered as Num\_P. The value of Num\_P can be 0 ~ 5, 8 ~ 10, 12 ~ 14, 17 ~ 28 and 30 ~ 31 (see Table 18).
- Interrupt\_I: stands for the CPU internal interrupt numbered as Num\_I. The value of Num\_I can be 6, 7, 11, 15, 16 and 29 (see Table 18).

#### 4.3.3.1 Allocate one peripheral interrupt source Source\_X to CPU

Setting the corresponding configuration register INTERRUPT\_PRO\_X\_MAP\_REG of Source\_X to Num\_P will allocate this interrupt source to Interrupt\_P. Num\_P here can be any value from 0 ~ 5, 8 ~ 10, 12 ~ 14, 17 ~ 28 and 30 ~ 31. Note that one CPU interrupt can be shared by multiple peripherals.

#### 4.3.3.2 Allocate multiple peripheral interrupt sources Source\_X<sub>n</sub> to CPU

Setting the corresponding configuration register INTERRUPT\_PRO\_X<sub>n</sub>\_MAP\_REG of each interrupt source to the same Num\_P will allocate all the sources to the same Interrupt\_P. Any of these sources will trigger CPU Interrupt\_P. When an interrupt signal is generated, software should check the interrupt status registers to figure out which peripheral the signal comes from.

#### 4.3.3.3 Disable CPU peripheral interrupt source Source\_X

Setting the corresponding configuration register INTERRUPT\_PRO\_X\_MAP\_REG of the source to any Num\_I will disable this interrupt Source\_X. The choice of Num\_I does not matter, as none of Num\_I is connected to the CPU. Therefore this functionality can be used to disable peripheral interrupt sources.

### 4.3.4 Disable CPU NMI Interrupt Sources

The interrupt matrix is able to mask all peripheral interrupt sources allocated to CPU No.14 NMI interrupt using hardware, depending on the internal signal [INTERRUPT\\_PRO\\_NMI\\_MASK\\_HW](#). The signal comes from "Interrupt Reg" register configuration submodule inside interrupt matrix, see Figure 4-1. If the signal is set to high level, CPU will not respond to NMI interrupt.

### 4.3.5 Query Current Interrupt Status of Peripheral Interrupt Source

Current interrupt status of a peripheral interrupt source can be read via the bit value in [INTERRUPT\\_PRO\\_INTR\\_STATUS\\_REG\\_n](#). For the mapping between [INTERRUPT\\_PRO\\_INTR\\_STATUS\\_REG\\_n](#) and peripheral interrupt sources, please refer to Table 17.

## 4.4 Base Address

Users can access interrupt matrix with the base address, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter 1: [System and Memory](#).

**Table 19: Interrupt Matrix Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F4C2000

## 4.5 Register Summary

The address in the following table represents the address offset (relative address) with the respect to the peripheral base address, not the absolute address. For detailed information about the interrupt matrix base address, please refer to Section 4.4.



Name	Description	Address	Access
<b>Configuration registers</b>			
<a href="#">INTERRUPT_PRO_MAC_INTR_MAP_REG</a>	MAC_INTR interrupt configuration register	0x0000	R/W
<a href="#">INTERRUPT_PRO_MAC_NMI_MAP_REG</a>	MAC_NMI interrupt configuration register	0x0004	R/W
<a href="#">INTERRUPT_PRO_PWR_INTR_MAP_REG</a>	PWR_INTR interrupt configuration register	0x0008	R/W
<a href="#">INTERRUPT_PRO_BB_INT_MAP_REG</a>	BB_INT interrupt configuration register	0x000C	R/W
<a href="#">INTERRUPT_PRO_BT_MAC_INT_MAP_REG</a>	BT_MAC_INT interrupt configuration register	0x0010	R/W
<a href="#">INTERRUPT_PRO_BT_BB_INT_MAP_REG</a>	BT_BB_INT interrupt configuration register	0x0014	R/W
<a href="#">INTERRUPT_PRO_BT_BB_NMI_MAP_REG</a>	BT_BB_NMI interrupt configuration register	0x0018	R/W
<a href="#">INTERRUPT_PRO_RWBT_IRQ_MAP_REG</a>	RWBT_IRQ interrupt configuration register	0x001C	R/W
<a href="#">INTERRUPT_PRO_RWBLE_IRQ_MAP_REG</a>	RWBLE_IRQ interrupt configuration register	0x0020	R/W
<a href="#">INTERRUPT_PRO_RWBT_NMI_MAP_REG</a>	RWBT_NMI interrupt configuration register	0x0024	R/W
<a href="#">INTERRUPT_PRO_RWBLE_NMI_MAP_REG</a>	RWBLE_NMI interrupt configuration register	0x0028	R/W
<a href="#">INTERRUPT_PRO_SLC0_INTR_MAP_REG</a>	SLC0_INTR interrupt configuration register	0x002C	R/W
<a href="#">INTERRUPT_PRO_SLC1_INTR_MAP_REG</a>	SLC1_INTR interrupt configuration register	0x0030	R/W
<a href="#">INTERRUPT_PRO_UHCI0_INTR_MAP_REG</a>	UHCI0_INTR interrupt configuration register	0x0034	R/W
<a href="#">INTERRUPT_PRO_UHCI1_INTR_MAP_REG</a>	UHCI1_INTR interrupt configuration register	0x0038	R/W
<a href="#">INTERRUPT_PRO_TG_T0_LEVEL_INT_MAP_REG</a>	TG_T0_LEVEL_INT interrupt configuration register	0x003C	R/W
<a href="#">INTERRUPT_PRO_TG_T1_LEVEL_INT_MAP_REG</a>	TG_T1_LEVEL_INT interrupt configuration register	0x0040	R/W
<a href="#">INTERRUPT_PRO_TG_WDT_LEVEL_INT_MAP_REG</a>	TG_WDT_LEVEL_INT interrupt configuration register	0x0044	R/W
<a href="#">INTERRUPT_PRO_TG_LACT_LEVEL_INT_MAP_REG</a>	TG_LACT_LEVEL_INT interrupt configuration register	0x0048	R/W
<a href="#">INTERRUPT_PRO_TG1_T0_LEVEL_INT_MAP_REG</a>	TG1_T0_LEVEL_INT interrupt configuration register	0x004C	R/W
<a href="#">INTERRUPT_PRO_TG1_T1_LEVEL_INT_MAP_REG</a>	TG1_T1_LEVEL_INT interrupt configuration register	0x0050	R/W
<a href="#">INTERRUPT_PRO_TG1_WDT_LEVEL_INT_MAP_REG</a>	TG1_WDT_LEVEL_INT interrupt configuration register	0x0054	R/W
<a href="#">INTERRUPT_PRO_TG1_LACT_LEVEL_INT_MAP_REG</a>	TG1_LACT_LEVEL_INT interrupt configuration register	0x0058	R/W
<a href="#">INTERRUPT_PRO_GPIO_INTERRUPT_PRO_MAP_REG</a>	GPIO_INTERRUPT_PRO interrupt configuration register	0x005C	R/W
<a href="#">INTERRUPT_PRO_GPIO_INTERRUPT_PRO_NMI_MAP_REG</a>	GPIO_INTERRUPT_PRO_NMI interrupt configuration register	0x0060	R/W
<a href="#">INTERRUPT_PRO_GPIO_INTERRUPT_APP_MAP_REG</a>	GPIO_INTERRUPT_APP interrupt configuration register	0x0064	R/W
<a href="#">INTERRUPT_PRO_GPIO_INTERRUPT_APP_NMI_MAP_REG</a>	GPIO_INTERRUPT_APP_NMI interrupt configuration register	0x0068	R/W
<a href="#">INTERRUPT_PRO_DEDICATED_GPIO_IN_INTR_MAP_REG</a>	DEDICATED_GPIO_IN_INTR interrupt configuration register	0x006C	R/W

Name	Description	Address	Access
<a href="#">INTERRUPT_PRO_CPU_INTR_FROM_CPU_0_MAP_REG</a>	CPU_INTR_FROM_CPU_0 interrupt configuration register	0x0070	R/W
<a href="#">INTERRUPT_PRO_CPU_INTR_FROM_CPU_1_MAP_REG</a>	CPU_INTR_FROM_CPU_1 interrupt configuration register	0x0074	R/W
<a href="#">INTERRUPT_PRO_CPU_INTR_FROM_CPU_2_MAP_REG</a>	CPU_INTR_FROM_CPU_2 interrupt configuration register	0x0078	R/W
<a href="#">INTERRUPT_PRO_CPU_INTR_FROM_CPU_3_MAP_REG</a>	CPU_INTR_FROM_CPU_3 interrupt configuration register	0x007C	R/W
<a href="#">INTERRUPT_PRO_SPI_INTR_1_MAP_REG</a>	SPI_INTR_1 interrupt configuration register	0x0080	R/W
<a href="#">INTERRUPT_PRO_SPI_INTR_2_MAP_REG</a>	SPI_INTR_2 interrupt configuration register	0x0084	R/W
<a href="#">INTERRUPT_PRO_SPI_INTR_3_MAP_REG</a>	SPI_INTR_3 interrupt configuration register	0x0088	R/W
<a href="#">INTERRUPT_PRO_I2S0_INT_MAP_REG</a>	I2S0_INT interrupt configuration register	0x008C	R/W
<a href="#">INTERRUPT_PRO_I2S1_INT_MAP_REG</a>	I2S1_INT interrupt configuration register	0x0090	R/W
<a href="#">INTERRUPT_PRO_UART_INTR_MAP_REG</a>	UART_INT interrupt configuration register	0x0094	R/W
<a href="#">INTERRUPT_PRO_UART1_INTR_MAP_REG</a>	UART1_INT interrupt configuration register	0x0098	R/W
<a href="#">INTERRUPT_PRO_UART2_INTR_MAP_REG</a>	UART2_INT interrupt configuration register	0x009C	R/W
<a href="#">INTERRUPT_PRO_SDIO_HOST_INTERRUPT_MAP_REG</a>	SDIO_HOST_INTERRUPT configuration register	0x00A0	R/W
<a href="#">INTERRUPT_PRO_PWM0_INTR_MAP_REG</a>	PWM0_INTR interrupt configuration register	0x00A4	R/W
<a href="#">INTERRUPT_PRO_PWM1_INTR_MAP_REG</a>	PWM1_INTR interrupt configuration register	0x00A8	R/W
<a href="#">INTERRUPT_PRO_PWM2_INTR_MAP_REG</a>	PWM2_INTR interrupt configuration register	0x00AC	R/W
<a href="#">INTERRUPT_PRO_PWM3_INTR_MAP_REG</a>	PWM3_INTR interrupt configuration register	0x00B0	R/W
<a href="#">INTERRUPT_PRO_LEDC_INT_MAP_REG</a>	LEDC_INTR interrupt configuration register	0x00B4	R/W
<a href="#">INTERRUPT_PRO_EFUSE_INT_MAP_REG</a>	EFUSE_INT interrupt configuration register	0x00B8	R/W
<a href="#">INTERRUPT_PRO_CAN_INT_MAP_REG</a>	CAN_INT interrupt configuration register	0x00BC	R/W
<a href="#">INTERRUPT_PRO_USB_INTR_MAP_REG</a>	USB_INT interrupt configuration register	0x00C0	R/W
<a href="#">INTERRUPT_PRO_RTC_CORE_INTR_MAP_REG</a>	RTC_CORE_INTR interrupt configuration register	0x00C4	R/W
<a href="#">INTERRUPT_PRO_RMT_INTR_MAP_REG</a>	RMT_INTR interrupt configuration register	0x00C8	R/W
<a href="#">INTERRUPT_PRO_PCNT_INTR_MAP_REG</a>	PCNT_INTR interrupt configuration register	0x00CC	R/W
<a href="#">INTERRUPT_PRO_I2C_EXT0_INTR_MAP_REG</a>	I2C_EXT0_INTR interrupt configuration register	0x00D0	R/W
<a href="#">INTERRUPT_PRO_I2C_EXT1_INTR_MAP_REG</a>	I2C_EXT1_INTR interrupt configuration register	0x00D4	R/W
<a href="#">INTERRUPT_PRO_RSA_INTR_MAP_REG</a>	RSA_INTR interrupt configuration register	0x00D8	R/W
<a href="#">INTERRUPT_PRO_SHA_INTR_MAP_REG</a>	SHA_INTR interrupt configuration register	0x00DC	R/W
<a href="#">INTERRUPT_PRO_AES_INTR_MAP_REG</a>	AES_INTR interrupt configuration register	0x00E0	R/W

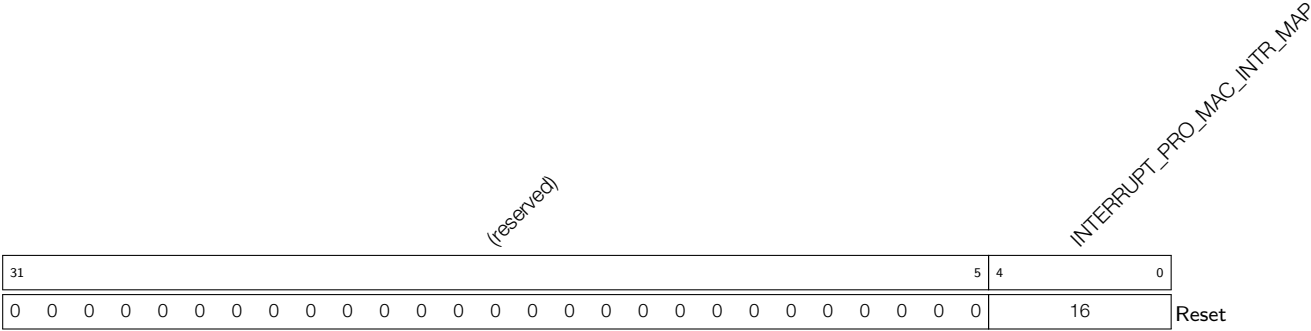
Name	Description	Address	Access
<a href="#">INTERRUPT_PRO_SPI2_DMA_INT_MAP_REG</a>	SPI2_DMA_INT interrupt configuration register	0x00E4	R/W
<a href="#">INTERRUPT_PRO_SPI3_DMA_INT_MAP_REG</a>	SPI3_DMA_INT interrupt configuration register	0x00E8	R/W
<a href="#">INTERRUPT_PRO_WDG_INT_MAP_REG</a>	WDG_INT interrupt configuration register	0x00EC	R/W
<a href="#">INTERRUPT_PRO_TIMER_INT1_MAP_REG</a>	TIMER_INT1 interrupt configuration register	0x00F0	R/W
<a href="#">INTERRUPT_PRO_TIMER_INT2_MAP_REG</a>	TIMER_INT2 interrupt configuration register	0x00F4	R/W
<a href="#">INTERRUPT_PRO_TG_T0_EDGE_INT_MAP_REG</a>	TG_T0_EDGE_INT interrupt configuration register	0x00F8	R/W
<a href="#">INTERRUPT_PRO_TG_T1_EDGE_INT_MAP_REG</a>	TG_T1_EDGE_INT interrupt configuration register	0x00FC	R/W
<a href="#">INTERRUPT_PRO_TG_WDT_EDGE_INT_MAP_REG</a>	TG_WDT_EDGE_INT interrupt configuration register	0x0100	R/W
<a href="#">INTERRUPT_PRO_TG_LACT_EDGE_INT_MAP_REG</a>	TG_LACT_EDGE_INT interrupt configuration register	0x0104	R/W
<a href="#">INTERRUPT_PRO_TG1_T0_EDGE_INT_MAP_REG</a>	TG1_T0_EDGE_INT interrupt configuration register	0x0108	R/W
<a href="#">INTERRUPT_PRO_TG1_T1_EDGE_INT_MAP_REG</a>	TG1_T1_EDGE_INT interrupt configuration register	0x010C	R/W
<a href="#">INTERRUPT_PRO_TG1_WDT_EDGE_INT_MAP_REG</a>	TG1_WDT_EDGE_INT interrupt configuration register	0x0110	R/W
<a href="#">INTERRUPT_PRO_TG1_LACT_EDGE_INT_MAP_REG</a>	TG1_LACT_EDGE_INT interrupt configuration register	0x0114	R/W
<a href="#">INTERRUPT_PRO_CACHE_IA_INT_MAP_REG</a>	CACHE_IA_INT interrupt configuration register	0x0118	R/W
<a href="#">INTERRUPT_PRO_SYSTIMER_TARGET0_INT_MAP_REG</a>	SYSTIMER_TARGET0_INT interrupt configuration register	0x011C	R/W
<a href="#">INTERRUPT_PRO_SYSTIMER_TARGET1_INT_MAP_REG</a>	SYSTIMER_TARGET1_INT interrupt configuration register	0x0120	R/W
<a href="#">INTERRUPT_PRO_SYSTIMER_TARGET2_INT_MAP_REG</a>	SYSTIMER_TARGET2 interrupt configuration register	0x0124	R/W
<a href="#">INTERRUPT_PRO_ASSIST_DEBUG_INTR_MAP_REG</a>	ASSIST_DEBUG_INTR interrupt configuration register	0x0128	R/W
<a href="#">INTERRUPT_PRO_PMS_PRO_IRAM0_ILG_INTR_MAP_REG</a>	PMS_PRO_IRAM0_ILG interrupt configuration register	0x012C	R/W
<a href="#">INTERRUPT_PRO_PMS_PRO_DRAM0_ILG_INTR_MAP_REG</a>	PMS_PRO_DRAM0_ILG interrupt configuration register	0x0130	R/W
<a href="#">INTERRUPT_PRO_PMS_PRO_DPORT_ILG_INTR_MAP_REG</a>	PMS_PRO_DPORT_ILG interrupt configuration register	0x0134	R/W
<a href="#">INTERRUPT_PRO_PMS_PRO_AHB_ILG_INTR_MAP_REG</a>	PMS_PRO_AHB_ILG interrupt configuration register	0x0138	R/W
<a href="#">INTERRUPT_PRO_PMS_PRO_CACHE_ILG_INTR_MAP_REG</a>	PMS_PRO_CACHE_ILG interrupt configuration register	0x013C	R/W
<a href="#">INTERRUPT_PRO_PMS_DMA_APB_I_ILG_INTR_MAP_REG</a>	PMS_DMA_APB_I_ILG interrupt configuration register	0x0140	R/W
<a href="#">INTERRUPT_PRO_PMS_DMA_RX_I_ILG_INTR_MAP_REG</a>	PMS_DMA_RX_I_ILG interrupt configuration register	0x0144	R/W
<a href="#">INTERRUPT_PRO_PMS_DMA_TX_I_ILG_INTR_MAP_REG</a>	PMS_DMA_TX_I_ILG interrupt configuration register	0x0148	R/W
<a href="#">INTERRUPT_PRO_SPI_MEM_REJECT_INTR_MAP_REG</a>	SPI_MEM_REJECT_INTR interrupt configuration register	0x014C	R/W
<a href="#">INTERRUPT_PRO_DMA_COPY_INTR_MAP_REG</a>	DMA_COPY_INTR interrupt configuration register	0x0150	R/W
<a href="#">INTERRUPT_PRO_SPI4_DMA_INT_MAP_REG</a>	SPI4_DMA_INT interrupt configuration register	0x0154	R/W

Name	Description	Address	Access
<a href="#">INTERRUPT_PRO_SPI_INTR_4_MAP_REG</a>	SPI_INTR_4 interrupt configuration register	0x0158	R/W
<a href="#">INTERRUPT_PRO_DCACHE_PRELOAD_INT_MAP_REG</a>	DCACHE_PRELOAD_INT interrupt configuration register	0x015C	R/W
<a href="#">INTERRUPT_PRO_ICACHE_PRELOAD_INT_MAP_REG</a>	ICACHE_PRELOAD_INT interrupt configuration register	0x0160	R/W
<a href="#">INTERRUPT_PRO_APB_ADC_INT_MAP_REG</a>	APB_ADC_INT interrupt configuration register	0x0164	R/W
<a href="#">INTERRUPT_PRO_CRYPTODMA_INT_MAP_REG</a>	CRYPTO_DMA_INT interrupt configuration register	0x0168	R/W
<a href="#">INTERRUPT_PRO_CPU_PERI_ERROR_INT_MAP_REG</a>	CPU_PERI_ERROR_INT interrupt configuration register	0x016C	R/W
<a href="#">INTERRUPT_PRO_APB_PERI_ERROR_INT_MAP_REG</a>	APB_PERI_ERROR_INT interrupt configuration register	0x0170	R/W
<a href="#">INTERRUPT_PRO_DCACHE_SYNC_INT_MAP_REG</a>	DCACHE_SYNC_INT interrupt configuration register	0x0174	R/W
<a href="#">INTERRUPT_PRO_ICACHE_SYNC_INT_MAP_REG</a>	ICACHE_SYNC_INT interrupt configuration register	0x0178	R/W
<a href="#">INTERRUPT_CLOCK_GATE_REG</a>	NMI interrupt signals mask register	0x0188	R/W
<b>Interrupt status registers</b>			
<a href="#">INTERRUPT_PRO_INTR_STATUS_REG_0_REG</a>	Interrupt status register 0	0x017C	RO
<a href="#">INTERRUPT_PRO_INTR_STATUS_REG_1_REG</a>	Interrupt status register 1	0x0180	RO
<a href="#">INTERRUPT_PRO_INTR_STATUS_REG_2_REG</a>	Interrupt status register 2	0x0184	RO
<b>Version register</b>			
<a href="#">INTERRUPT_DATE_REG</a>	Version control register	0x0FFC	R/W

4.6 Registers

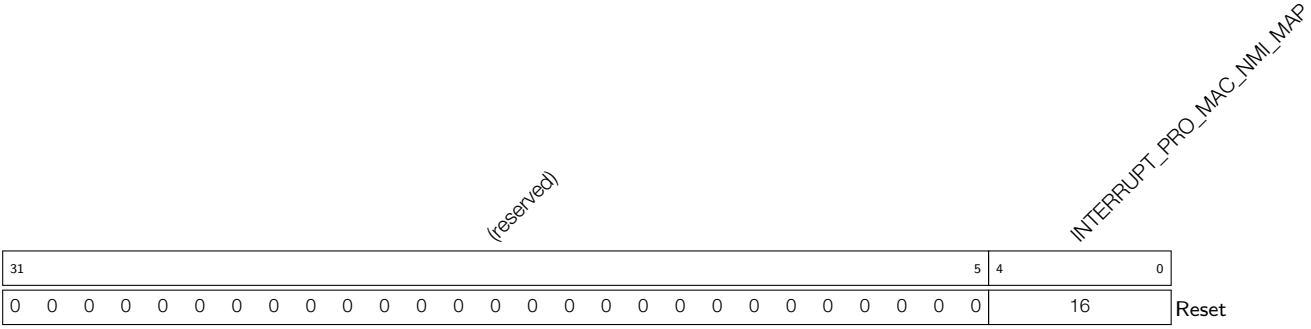
The address in the following part represents the address offset (relative address) with respect to the peripheral base address, not the absolute address. For detailed information about the interrupt matrix base address, please refer to Section 4.4.

Register 4.1: INTERRUPT\_PRO\_MAC\_INTR\_MAP\_REG (0x0000)



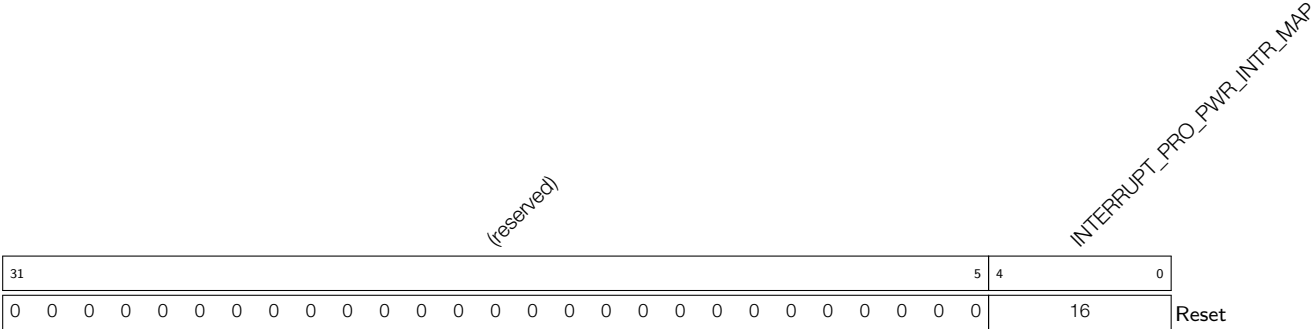
**INTERRUPT\_PRO\_MAC\_INTR\_MAP** This register is used to map MAC\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.2: INTERRUPT\_PRO\_MAC\_NMI\_MAP\_REG (0x0004)



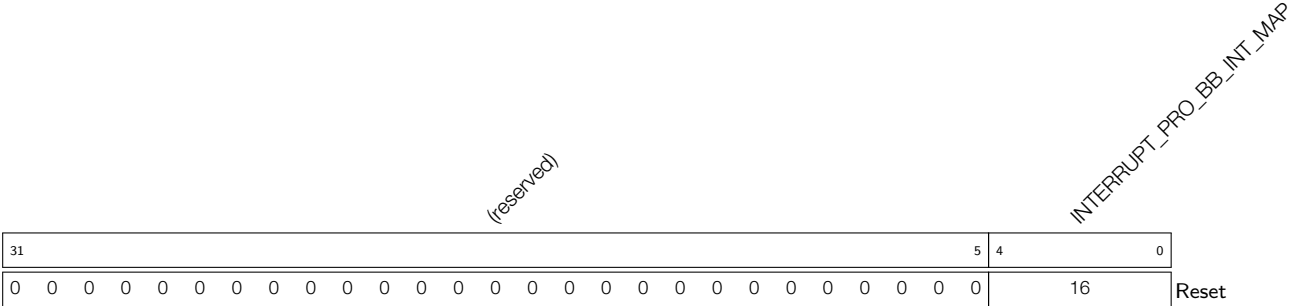
**INTERRUPT\_PRO\_MAC\_NMI\_MAP** This register is used to map MAC\_NMI interrupt signal to one of the CPU interrupts. (R/W)

Register 4.3: INTERRUPT\_PRO\_PWR\_INTR\_MAP\_REG (0x0008)



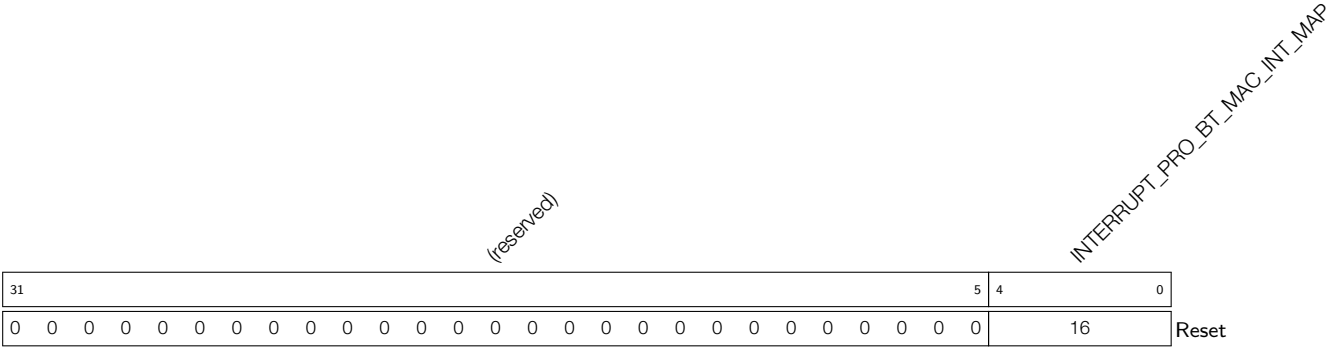
**INTERRUPT\_PRO\_PWR\_INTR\_MAP** This register is used to map PWR\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.4: INTERRUPT\_PRO\_BB\_INT\_MAP\_REG (0x000C)



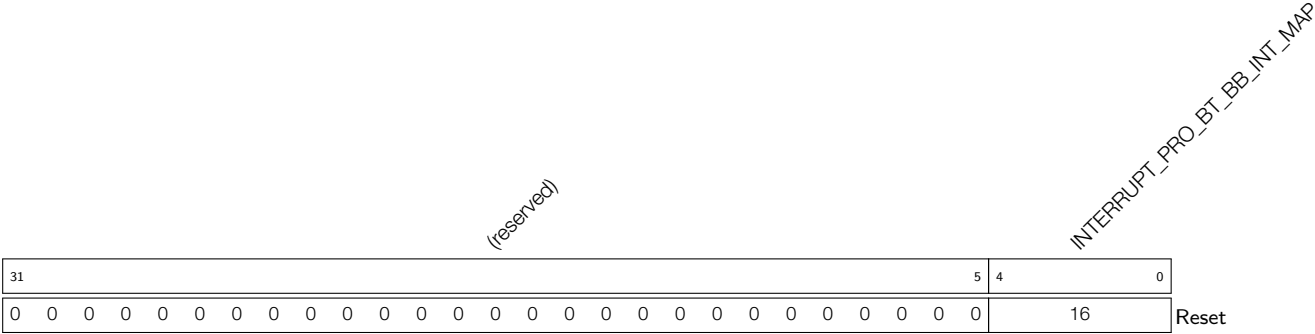
**INTERRUPT\_PRO\_BB\_INT\_MAP** This register is used to map BB\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.5: INTERRUPT\_PRO\_BT\_MAC\_INT\_MAP\_REG (0x0010)



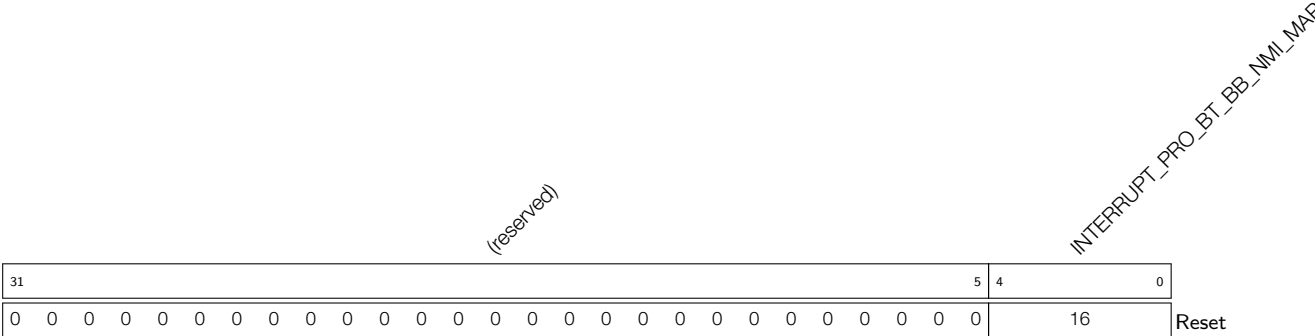
**INTERRUPT\_PRO\_BT\_MAC\_INT\_MAP** This register is used to map BT\_MAC\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.6: INTERRUPT\_PRO\_BT\_BB\_INT\_MAP\_REG (0x0014)



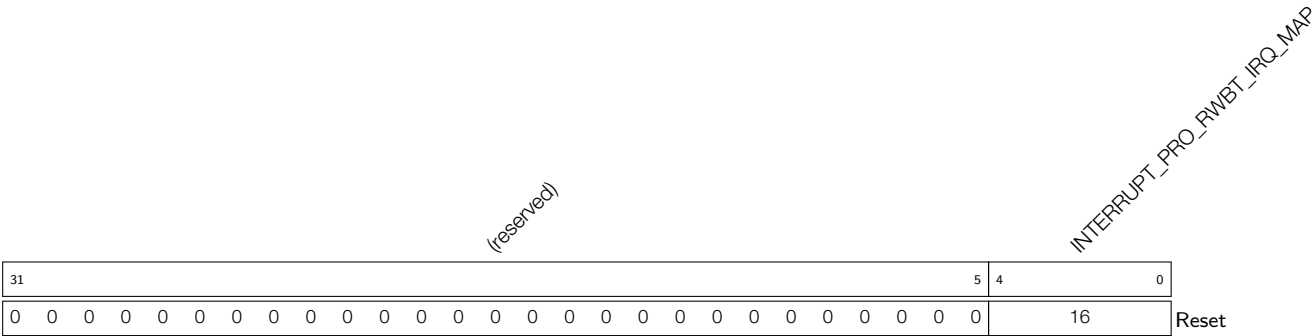
**INTERRUPT\_PRO\_BT\_BB\_INT\_MAP** This register is used to map BT\_BB\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.7: INTERRUPT\_PRO\_BT\_BB\_NMI\_MAP\_REG (0x0018)



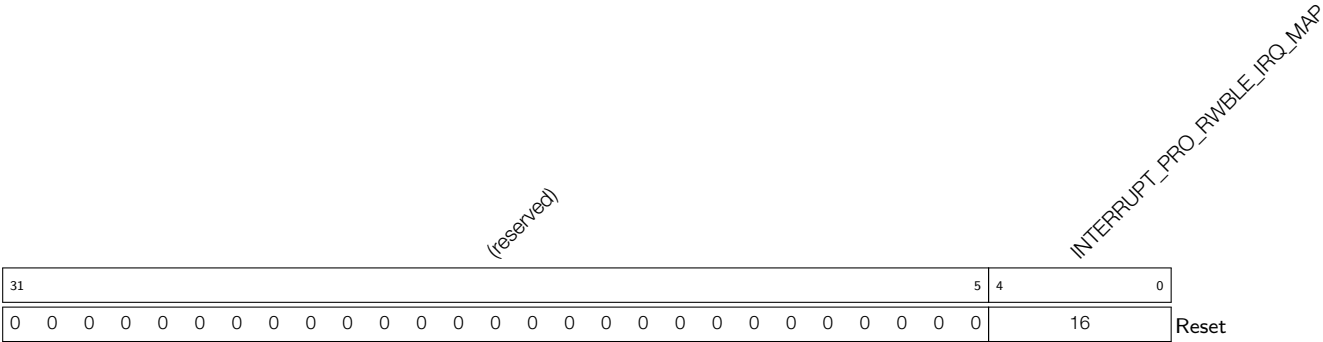
**INTERRUPT\_PRO\_BT\_BB\_NMI\_MAP** This register is used to map BT\_BB\_NMI interrupt signal to one of the CPU interrupts. (R/W)

Register 4.8: INTERRUPT\_PRO\_RWBT\_IRQ\_MAP\_REG (0x001C)



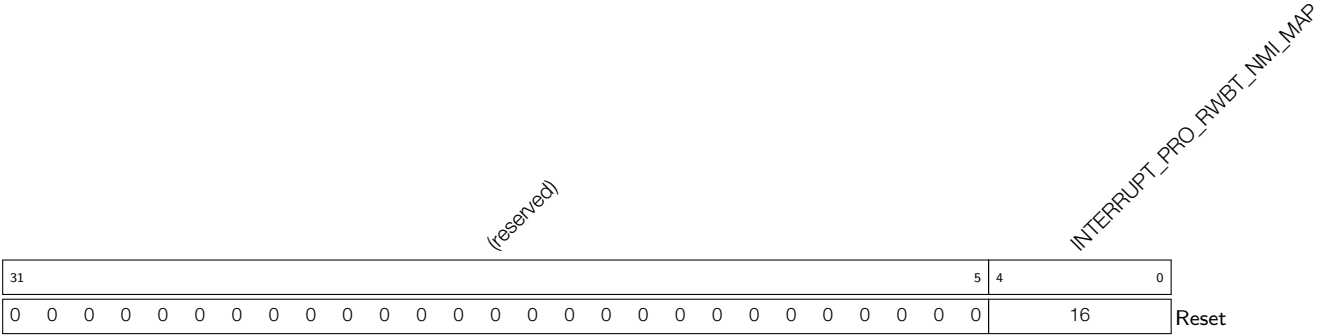
**INTERRUPT\_PRO\_RWBT\_IRQ\_MAP** This register is used to map RWBT\_IRQ interrupt signal to one of the CPU interrupts. (R/W)

Register 4.9: INTERRUPT\_PRO\_RWBLE\_IRQ\_MAP\_REG (0x0020)



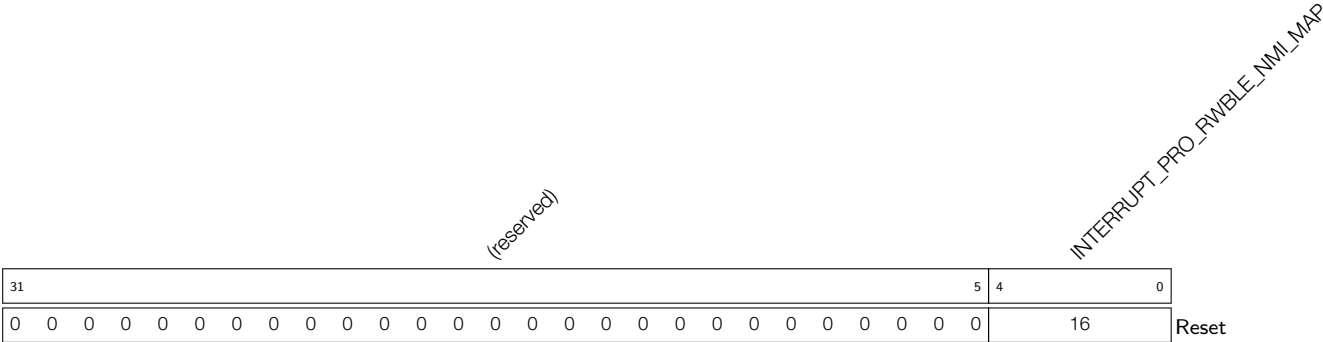
**INTERRUPT\_PRO\_RWBLE\_IRQ\_MAP** This register is used to map RWBLE\_IRQ interrupt signal to one of the CPU interrupts. (R/W)

Register 4.10: INTERRUPT\_PRO\_RWBT\_NMI\_MAP\_REG (0x0024)



**INTERRUPT\_PRO\_RWBT\_NMI\_MAP** This register is used to map RWBT\_NMI interrupt signal to one of the CPU interrupts. (R/W)

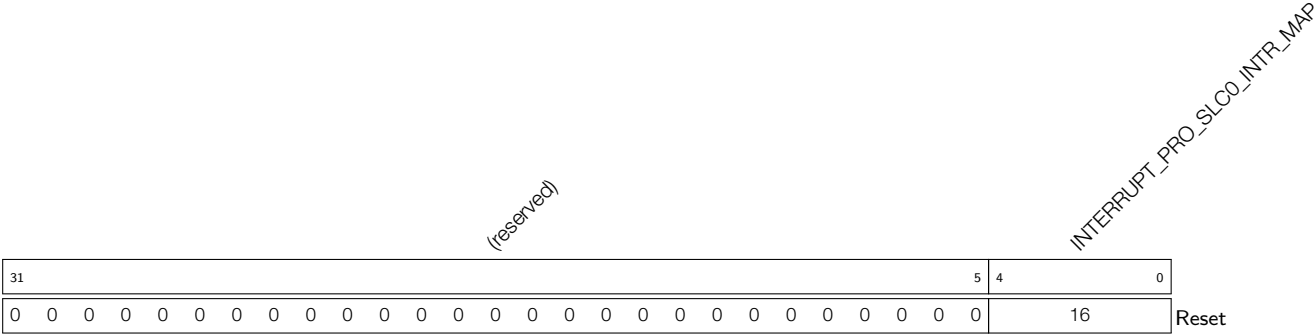
Register 4.11: INTERRUPT\_PRO\_RWBLE\_NMI\_MAP\_REG (0x0028)



**INTERRUPT\_PRO\_RWBLE\_NMI\_MAP** This register is used to map RWBLE\_NMI interrupt signal to one of the CPU interrupts. (R/W)

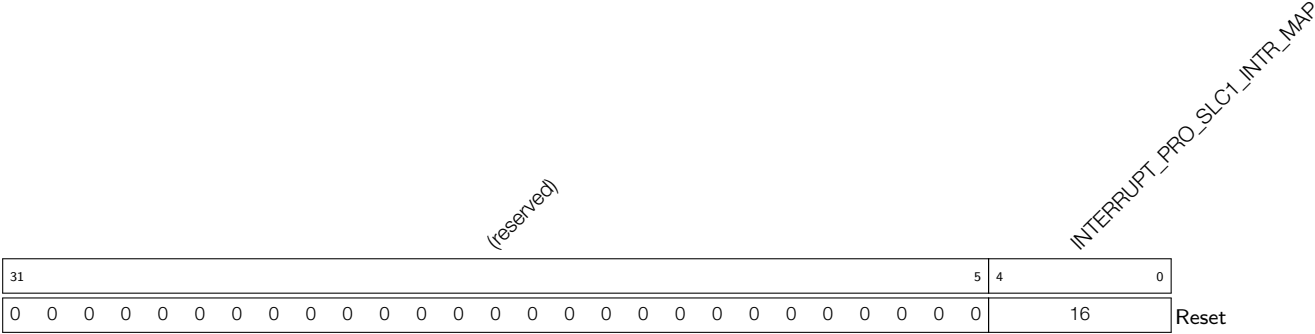


Register 4.12: INTERRUPT\_PRO\_SLC0\_INTR\_MAP\_REG (0x002C)



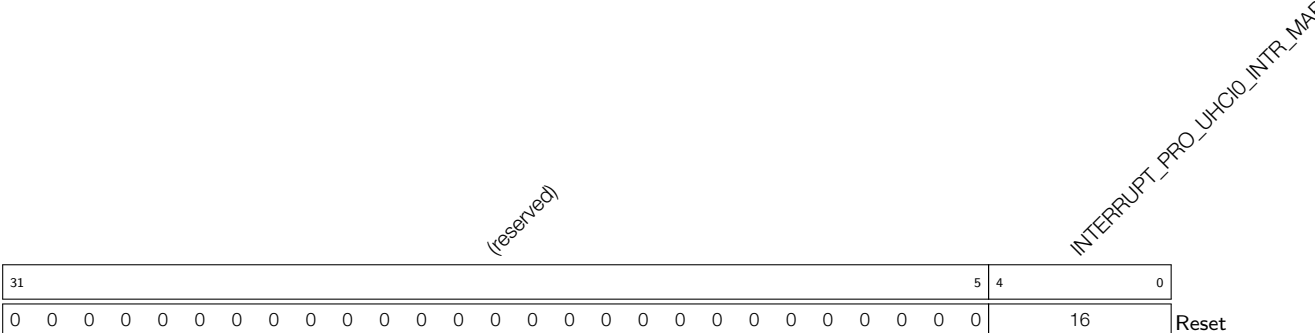
**INTERRUPT\_PRO\_SLC0\_INTR\_MAP** This register is used to map SLC0\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.13: INTERRUPT\_PRO\_SLC1\_INTR\_MAP\_REG (0x0030)



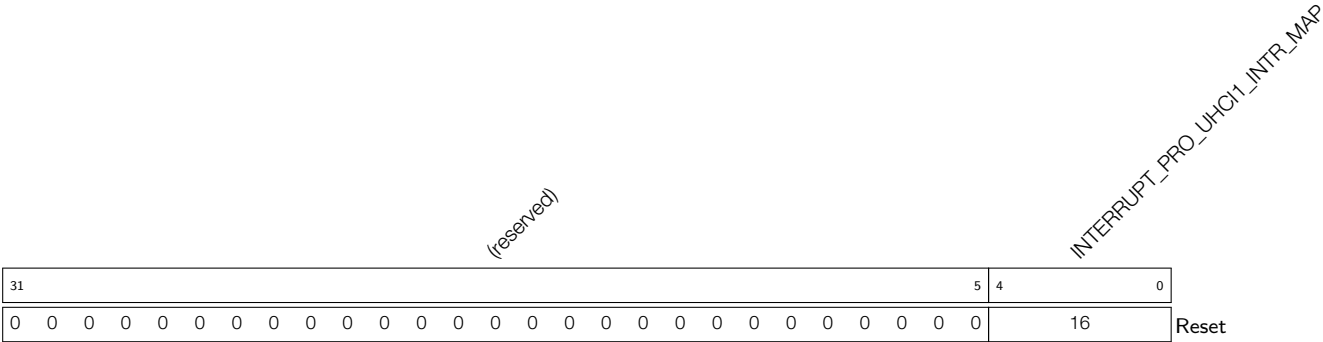
**INTERRUPT\_PRO\_SLC1\_INTR\_MAP** This register is used to map SLC1\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.14: INTERRUPT\_PRO\_UHCI0\_INTR\_MAP\_REG (0x0034)



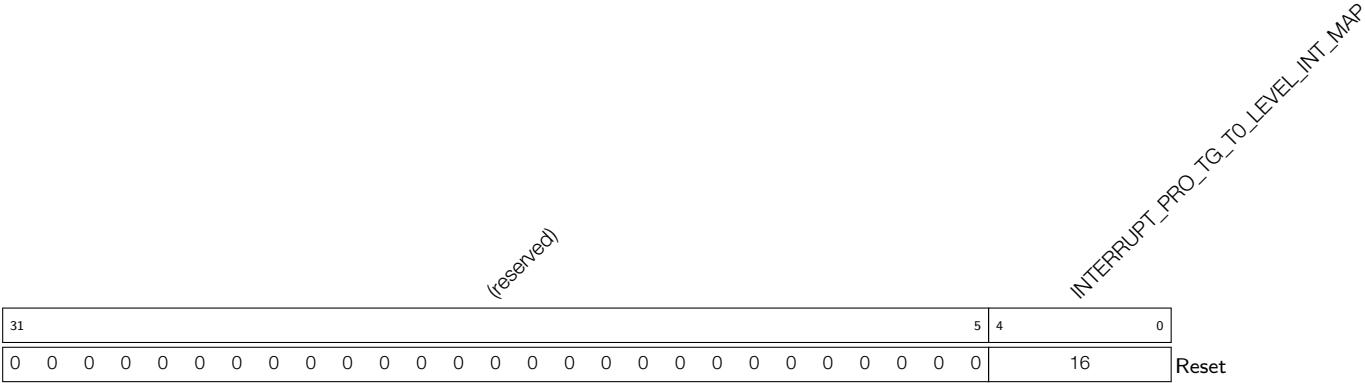
**INTERRUPT\_PRO\_UHCI0\_INTR\_MAP** This register is used to map UHCI0\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.15: INTERRUPT\_PRO\_UHCI1\_INTR\_MAP\_REG (0x0038)



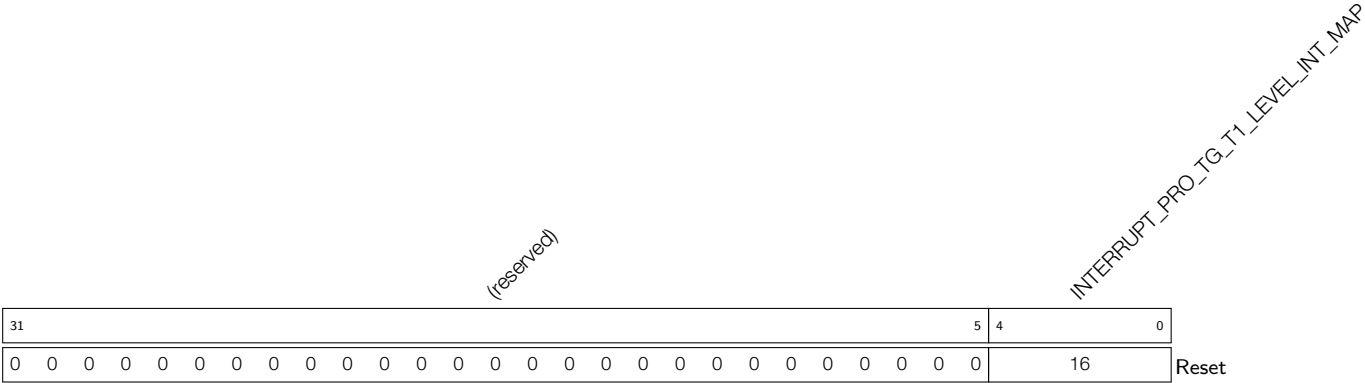
**INTERRUPT\_PRO\_UHCI1\_INTR\_MAP** This register is used to map UHCI1\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.16: INTERRUPT\_PRO\_TG\_T0\_LEVEL\_INT\_MAP\_REG (0x003C)



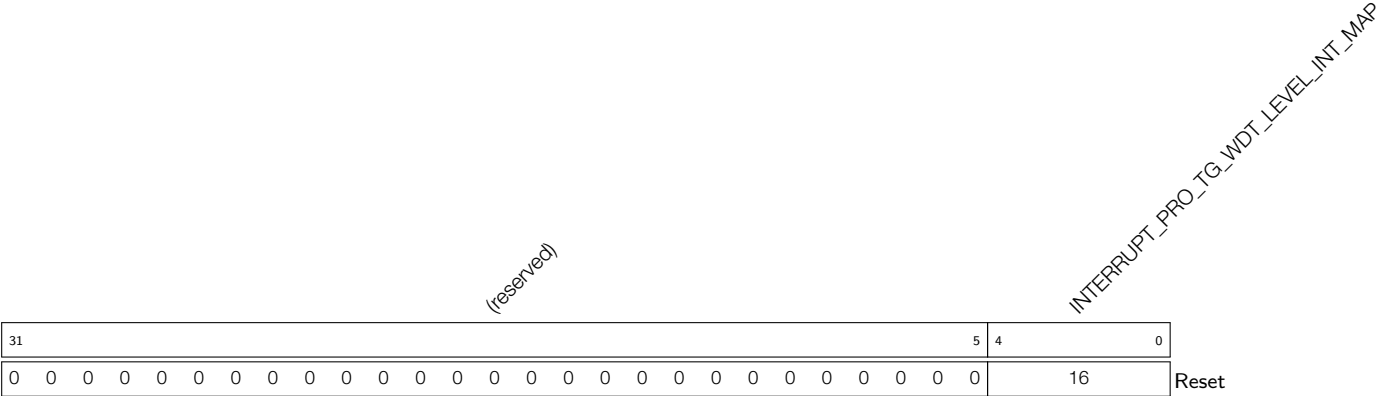
**INTERRUPT\_PRO\_TG\_T0\_LEVEL\_INT\_MAP** This register is used to map TG\_T0\_LEVEL\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.17: INTERRUPT\_PRO\_TG\_T1\_LEVEL\_INT\_MAP\_REG (0x0040)



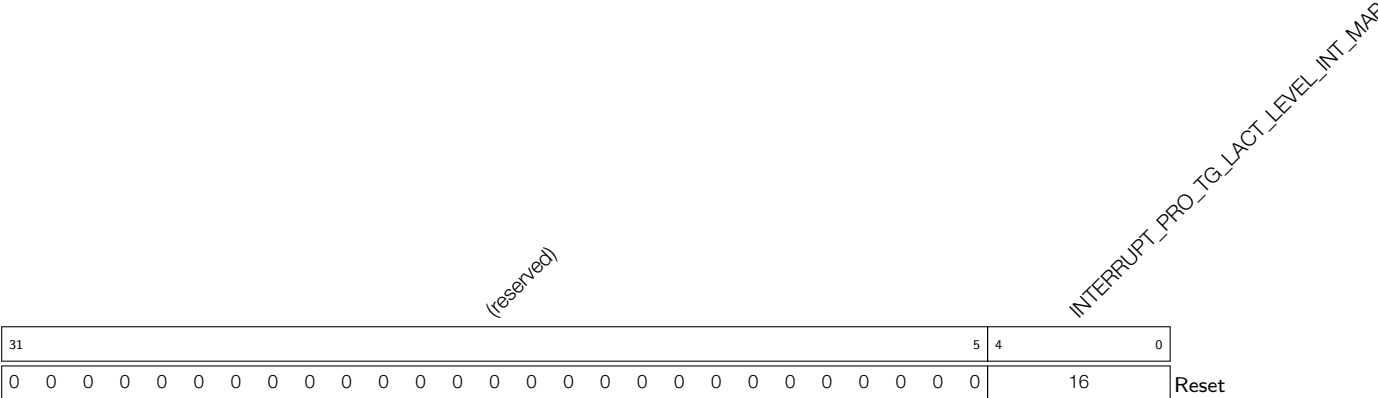
**INTERRUPT\_PRO\_TG\_T1\_LEVEL\_INT\_MAP** This register is used to map TG\_T1\_LEVEL\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.18: INTERRUPT\_PRO\_TG\_WDT\_LEVEL\_INT\_MAP\_REG (0x0044)



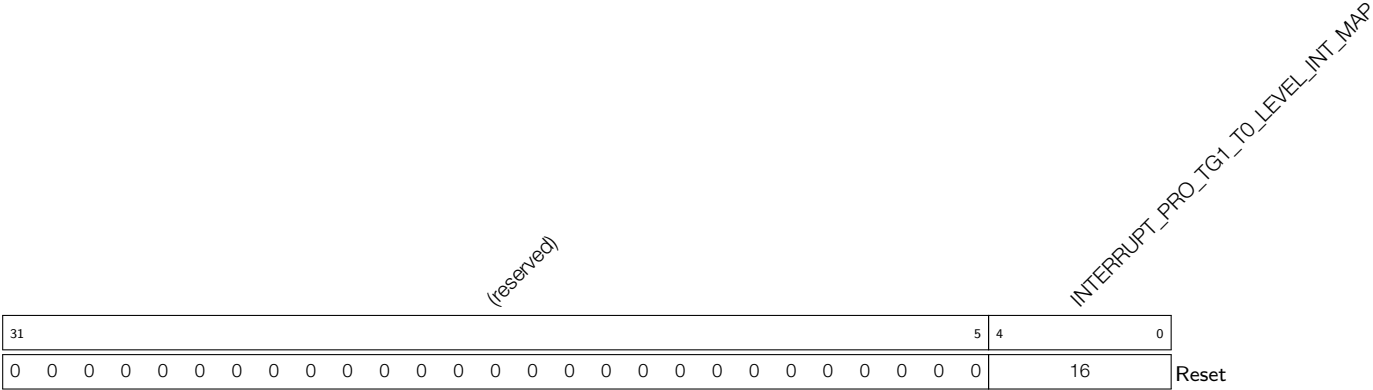
**INTERRUPT\_PRO\_TG\_WDT\_LEVEL\_INT\_MAP** This register is used to map TG\_WDT\_LEVEL\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.19: INTERRUPT\_PRO\_TG\_LACT\_LEVEL\_INT\_MAP\_REG (0x0048)



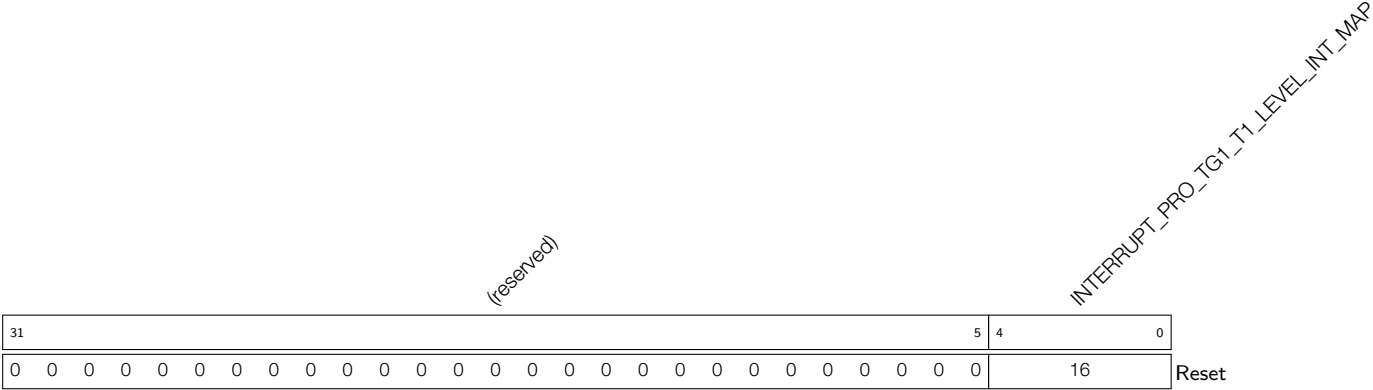
**INTERRUPT\_PRO\_TG\_LACT\_LEVEL\_INT\_MAP** This register is used to map TG\_LACT\_LEVEL\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.20: INTERRUPT\_PRO\_TG1\_T0\_LEVEL\_INT\_MAP\_REG (0x004C)



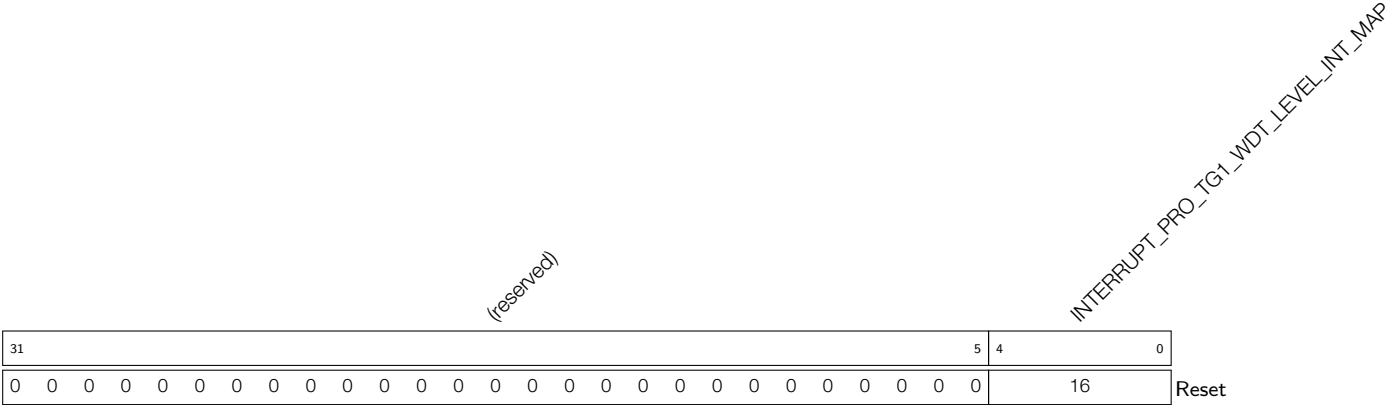
**INTERRUPT\_PRO\_TG1\_T0\_LEVEL\_INT\_MAP** This register is used to map TG1\_T0\_LEVEL\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.21: INTERRUPT\_PRO\_TG1\_T1\_LEVEL\_INT\_MAP\_REG (0x0050)



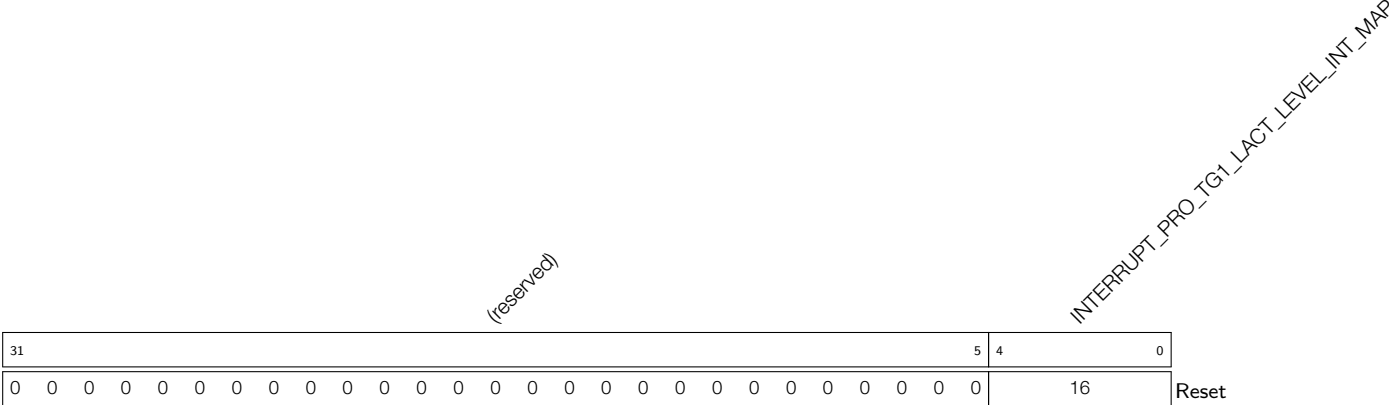
**INTERRUPT\_PRO\_TG1\_T1\_LEVEL\_INT\_MAP** This register is used to map TG1\_T1\_LEVEL\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.22: INTERRUPT\_PRO\_TG1\_WDT\_LEVEL\_INT\_MAP\_REG (0x0054)



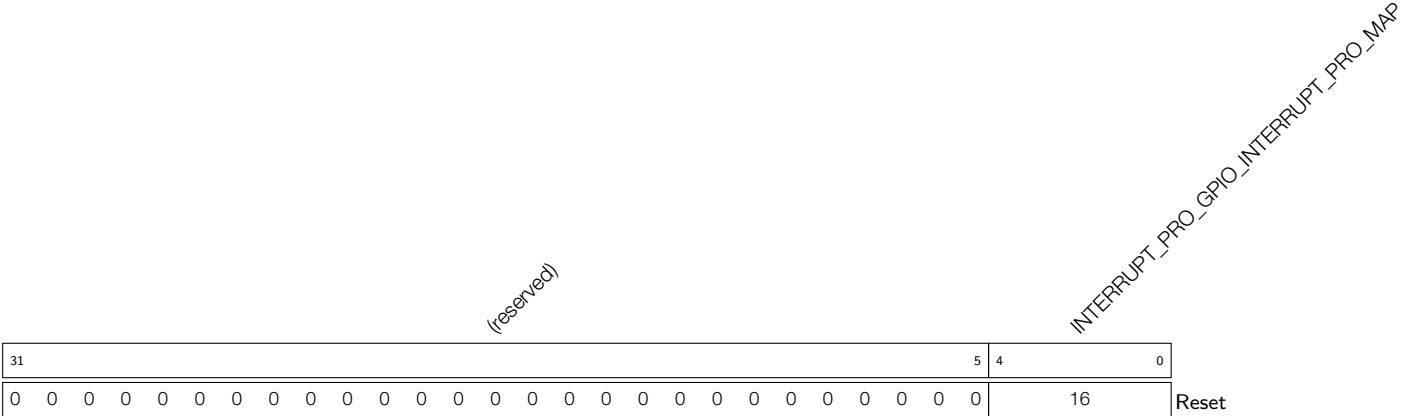
**INTERRUPT\_PRO\_TG1\_WDT\_LEVEL\_INT\_MAP** This register is used to map TG1\_WDT\_LEVEL\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.23: INTERRUPT\_PRO\_TG1\_LACT\_LEVEL\_INT\_MAP\_REG (0x0058)



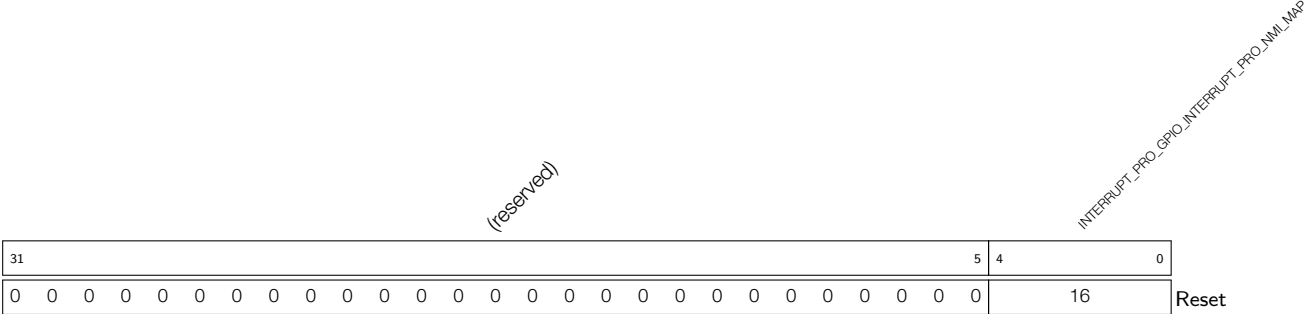
**INTERRUPT\_PRO\_TG1\_LACT\_LEVEL\_INT\_MAP** This register is used to map TG1\_LACT\_LEVEL\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.24: INTERRUPT\_PRO\_GPIO\_INTERRUPT\_PRO\_MAP\_REG (0x005C)



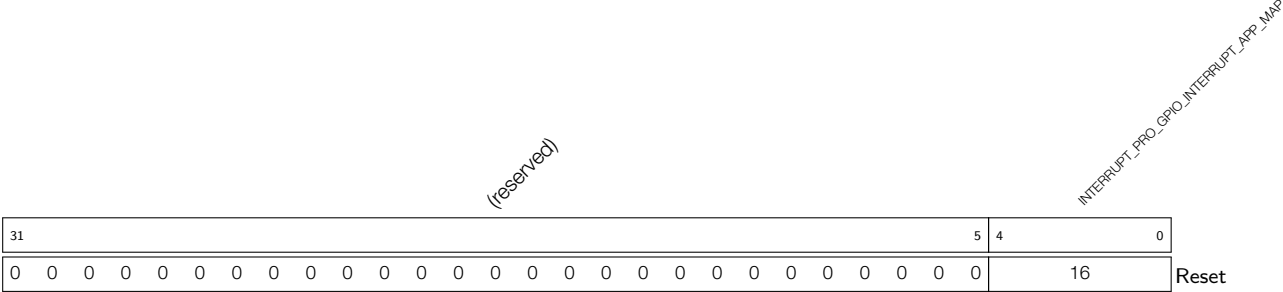
**INTERRUPT\_PRO\_GPIO\_INTERRUPT\_PRO\_MAP** This register is used to map GPIO\_INTERRUPT\_PRO interrupt signal to one of the CPU interrupts. (R/W)

Register 4.25: INTERRUPT\_PRO\_GPIO\_INTERRUPT\_PRO\_NMI\_MAP\_REG (0x0060)



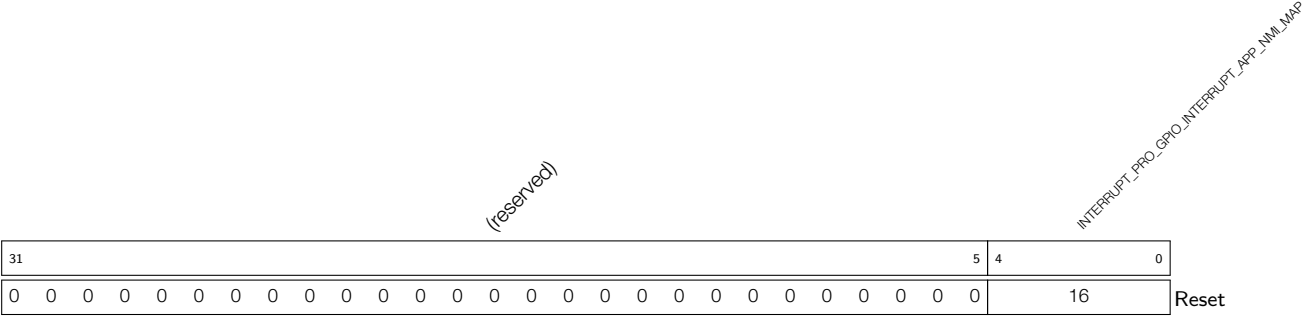
**INTERRUPT\_PRO\_GPIO\_INTERRUPT\_PRO\_NMI\_MAP** This register is used to map GPIO\_INTERRUPT\_PRO\_NMI interrupt signal to one of the CPU interrupts. (R/W)

Register 4.26: INTERRUPT\_PRO\_GPIO\_INTERRUPT\_APP\_MAP\_REG (0x0064)



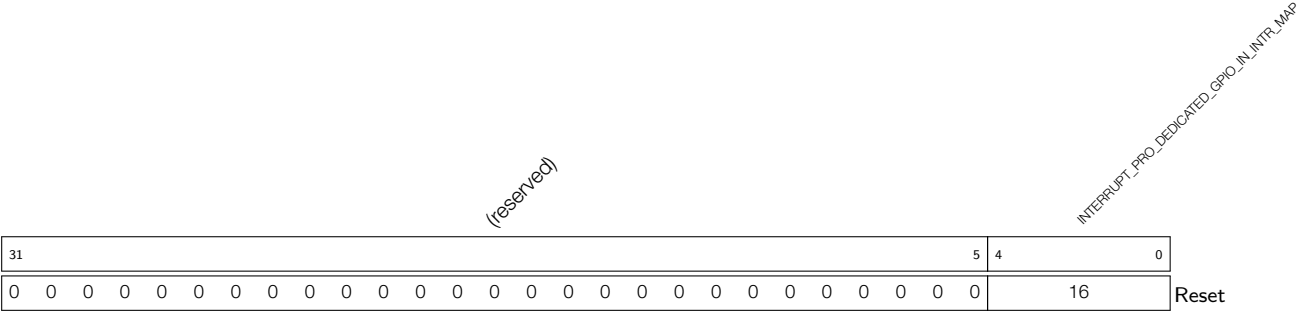
**INTERRUPT\_PRO\_GPIO\_INTERRUPT\_APP\_MAP** This register is used to map GPIO\_INTERRUPT\_APP interrupt signal to one of the CPU interrupts. (R/W)

Register 4.27: INTERRUPT\_PRO\_GPIO\_INTERRUPT\_APP\_NMI\_MAP\_REG (0x0068)



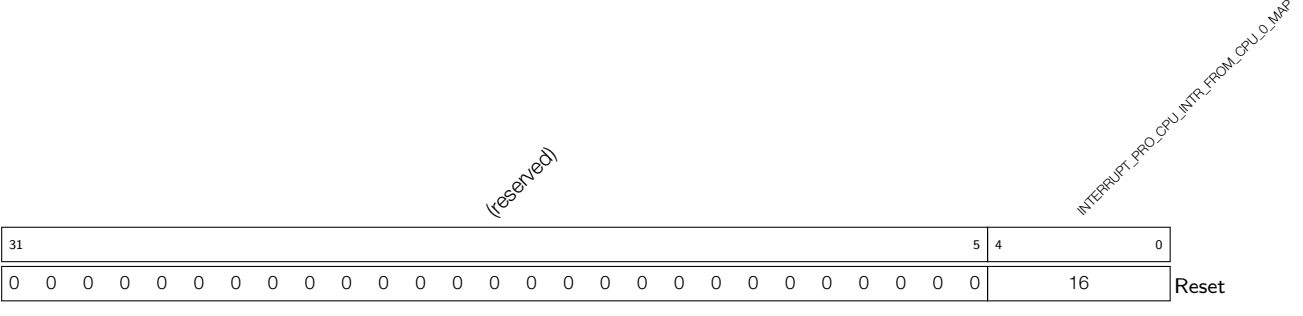
**INTERRUPT\_PRO\_GPIO\_INTERRUPT\_APP\_NMI\_MAP** This register is used to map GPIO\_INTERRUPT\_APP\_NMI interrupt signal to one of the CPU interrupts. (R/W)

Register 4.28: INTERRUPT\_PRO\_DEDICATED\_GPIO\_IN\_INTR\_MAP\_REG (0x006C)



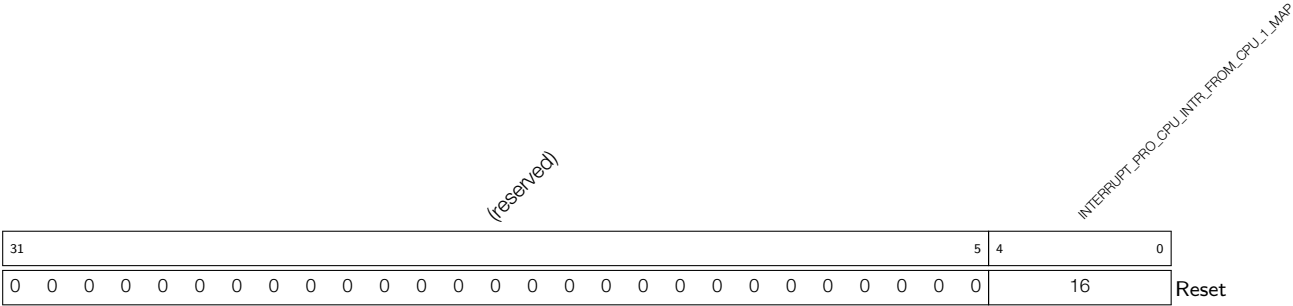
**INTERRUPT\_PRO\_DEDICATED\_GPIO\_IN\_INTR\_MAP** This register is used to map DEDICATED\_GPIO\_IN\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.29: INTERRUPT\_PRO\_CPU\_INTR\_FROM\_CPU\_0\_MAP\_REG (0x0070)



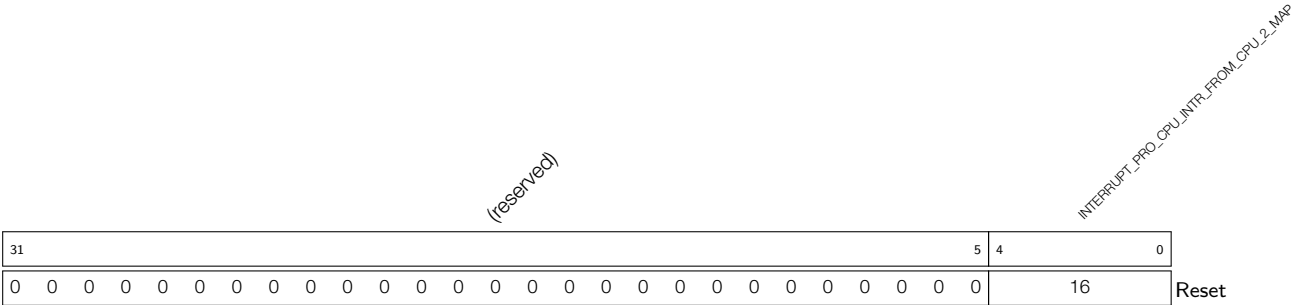
**INTERRUPT\_PRO\_CPU\_INTR\_FROM\_CPU\_0\_MAP** This register is used to map CPU\_INTR\_FROM\_CPU\_0 interrupt signal to one of the CPU interrupts. (R/W)

Register 4.30: INTERRUPT\_PRO\_CPU\_INTR\_FROM\_CPU\_1\_MAP\_REG (0x0074)



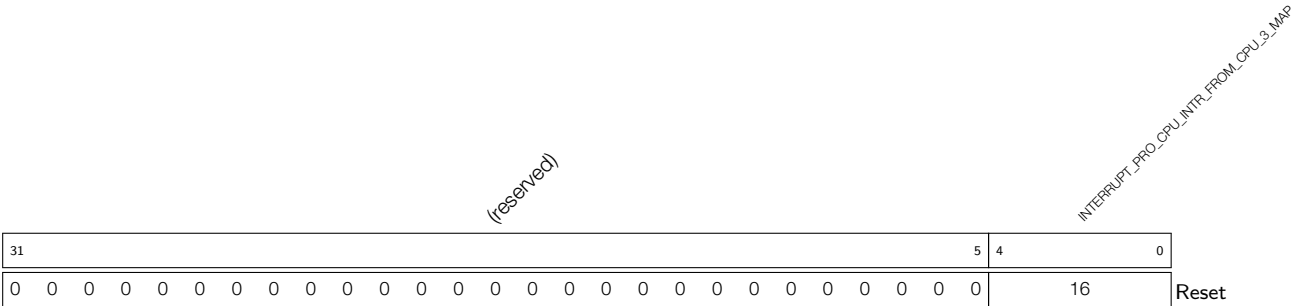
**INTERRUPT\_PRO\_CPU\_INTR\_FROM\_CPU\_1\_MAP** This register is used to map CPU\_INTR\_FROM\_CPU\_1 interrupt signal to one of the CPU interrupts. (R/W)

Register 4.31: INTERRUPT\_PRO\_CPU\_INTR\_FROM\_CPU\_2\_MAP\_REG (0x0078)



**INTERRUPT\_PRO\_CPU\_INTR\_FROM\_CPU\_2\_MAP** This register is used to map CPU\_INTR\_FROM\_CPU\_2 interrupt signal to one of the CPU interrupts. (R/W)

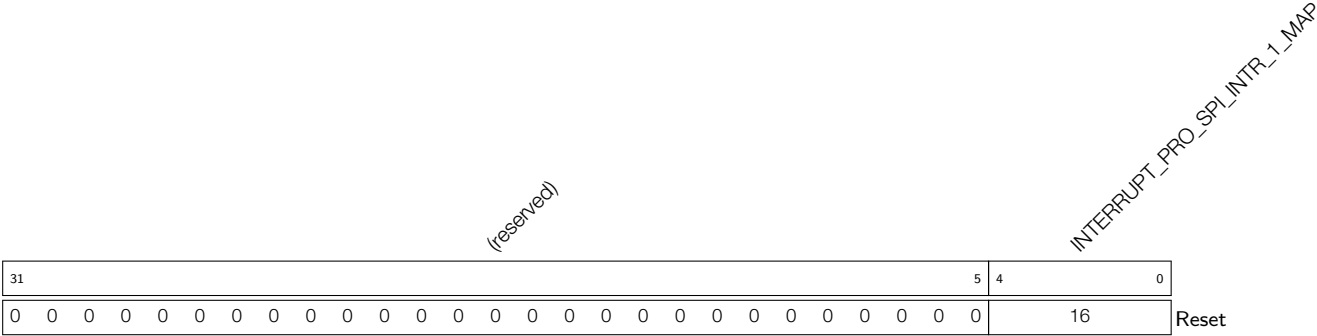
Register 4.32: INTERRUPT\_PRO\_CPU\_INTR\_FROM\_CPU\_3\_MAP\_REG (0x007C)



**INTERRUPT\_PRO\_CPU\_INTR\_FROM\_CPU\_3\_MAP** This register is used to map CPU\_INTR\_FROM\_CPU\_3 interrupt signal to one of the CPU interrupts. (R/W)

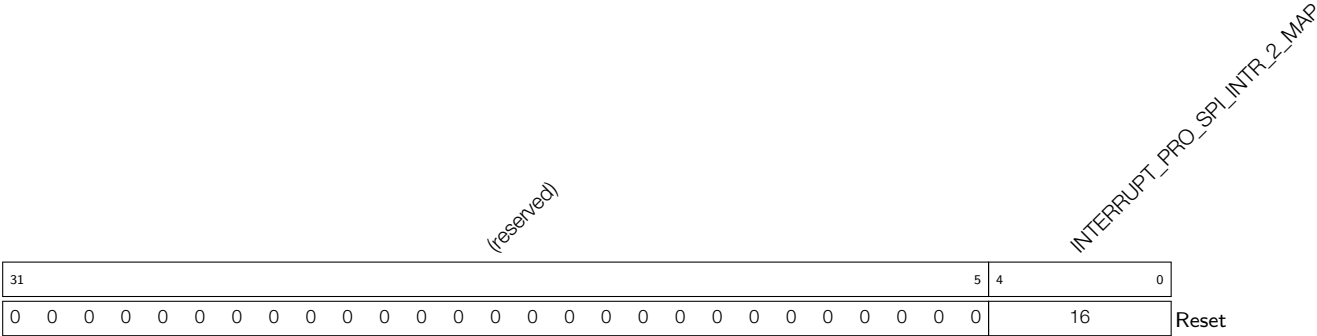


Register 4.33: INTERRUPT\_PRO\_SPI\_INTR\_1\_MAP\_REG (0x0080)



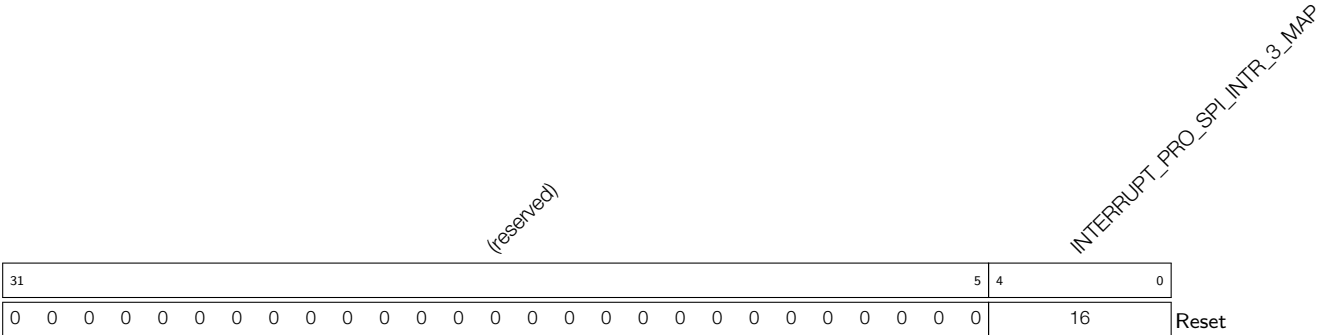
**INTERRUPT\_PRO\_SPI\_INTR\_1\_MAP** This register is used to map SPI\_INTR\_1 interrupt signal to one of the CPU interrupts. (R/W)

Register 4.34: INTERRUPT\_PRO\_SPI\_INTR\_2\_MAP\_REG (0x0084)

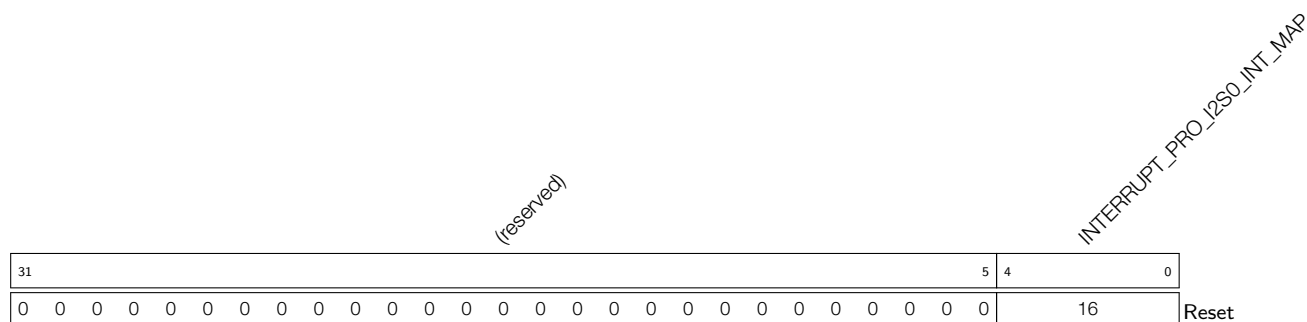


**INTERRUPT\_PRO\_SPI\_INTR\_2\_MAP** This register is used to map SPI\_INTR\_2 interrupt signal to one of the CPU interrupts. (R/W)

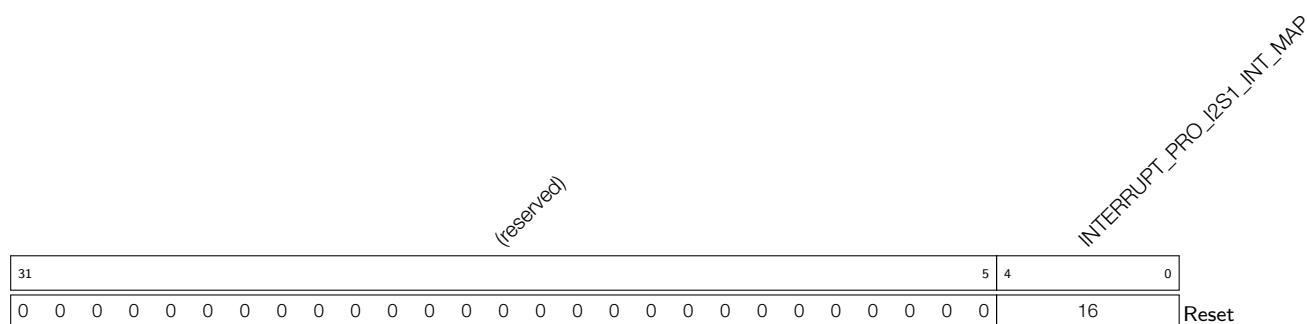
Register 4.35: INTERRUPT\_PRO\_SPI\_INTR\_3\_MAP\_REG (0x0088)



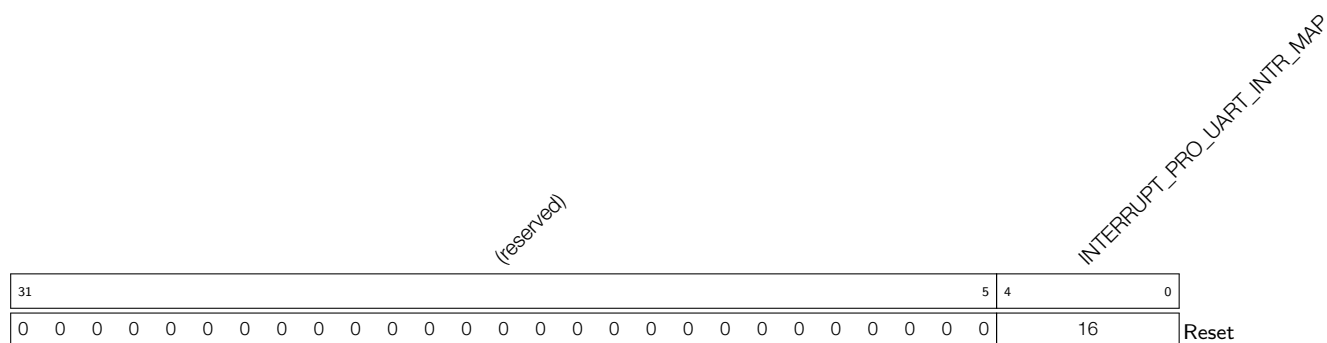
**INTERRUPT\_PRO\_SPI\_INTR\_3\_MAP** This register is used to map SPI\_INTR\_3 interrupt signal to one of the CPU interrupts. (R/W)

**Register 4.36: INTERRUPT\_PRO\_I2S0\_INT\_MAP\_REG (0x008C)**

**INTERRUPT\_PRO\_I2S0\_INT\_MAP** This register is used to map I2S0\_INT interrupt signal to one of the CPU interrupts. (R/W)

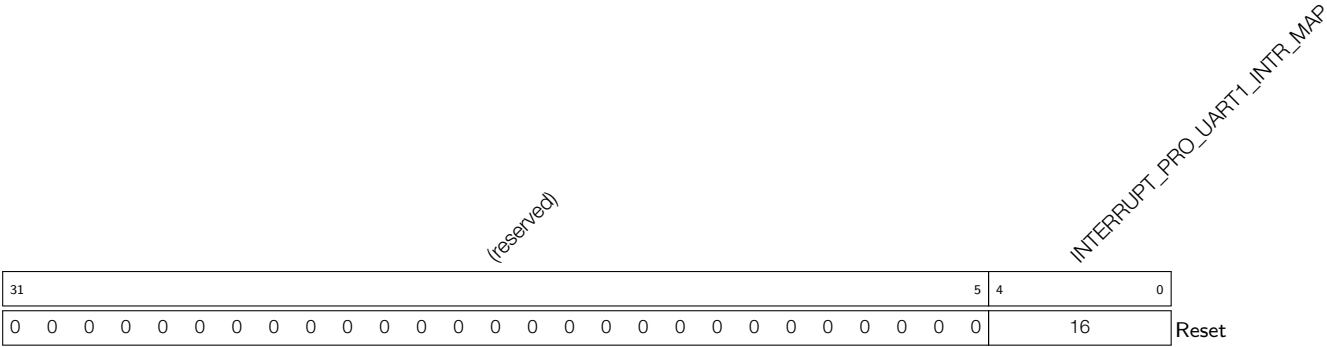
**Register 4.37: INTERRUPT\_PRO\_I2S1\_INT\_MAP\_REG (0x0090)**

**INTERRUPT\_PRO\_I2S1\_INT\_MAP** This register is used to map I2S1\_INT interrupt signal to one of the CPU interrupts. (R/W)

**Register 4.38: INTERRUPT\_PRO\_UART\_INTR\_MAP\_REG (0x0094)**

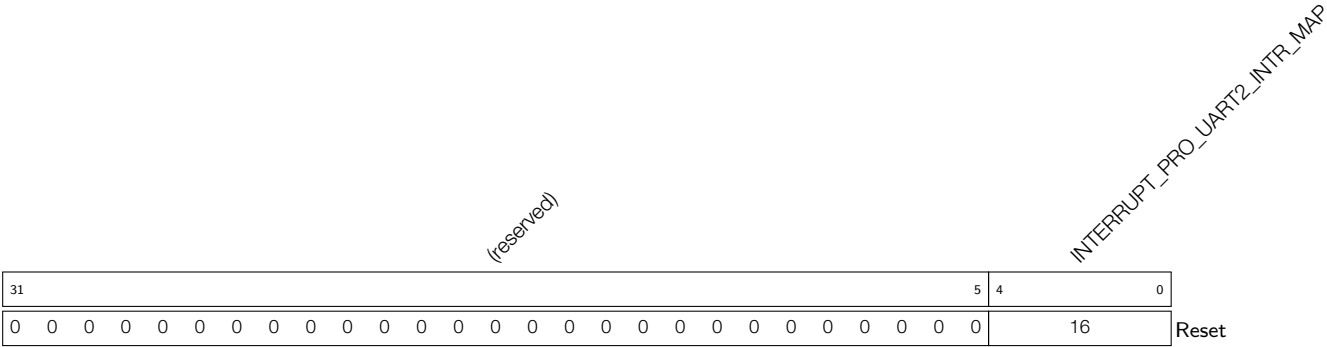
**INTERRUPT\_PRO\_UART\_INTR\_MAP** This register is used to map UART\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.39: INTERRUPT\_PRO\_UART1\_INTR\_MAP\_REG (0x0098)



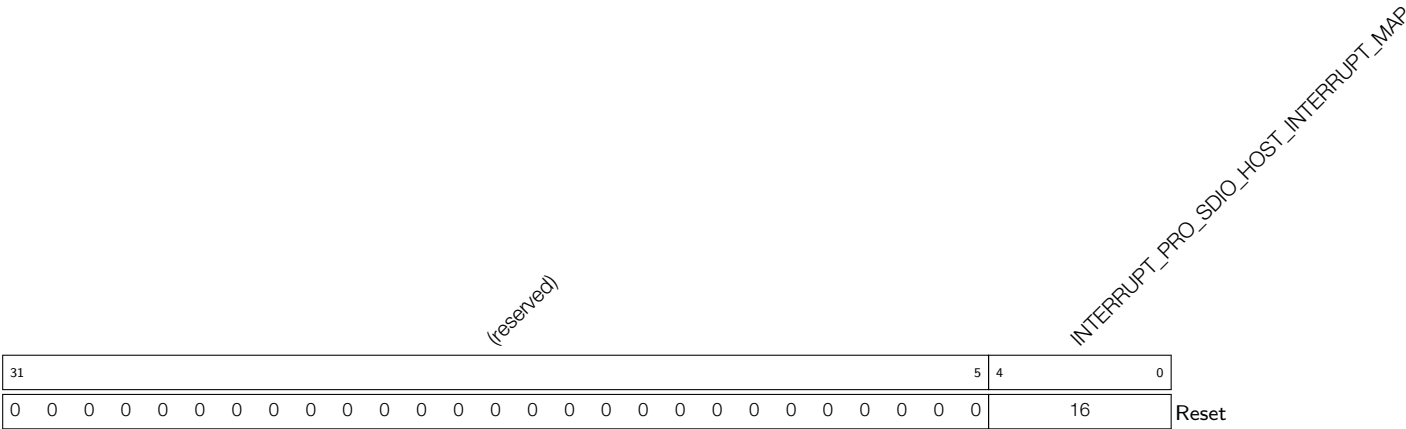
**INTERRUPT\_PRO\_UART1\_INTR\_MAP** This register is used to map UART1\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.40: INTERRUPT\_PRO\_UART2\_INTR\_MAP\_REG (0x009C)



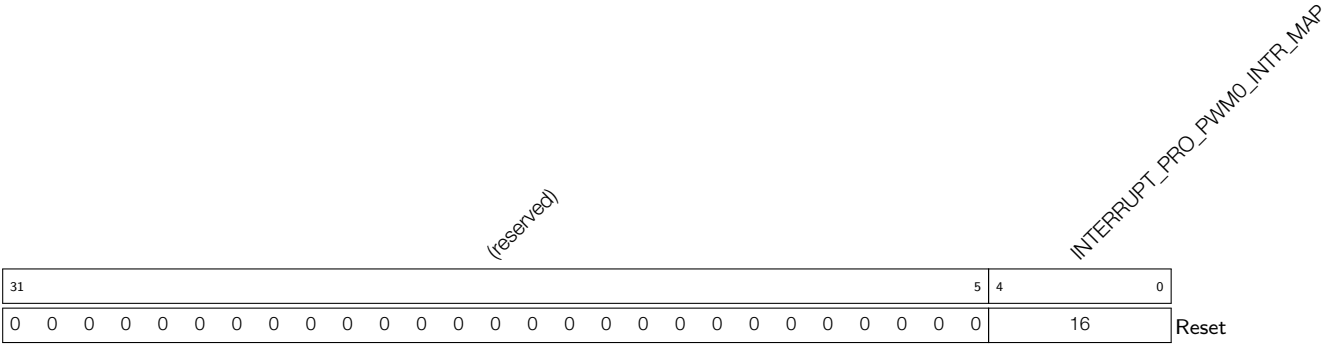
**INTERRUPT\_PRO\_UART2\_INTR\_MAP** This register is used to map UART2\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.41: INTERRUPT\_PRO\_SDIO\_HOST\_INTERRUPT\_MAP\_REG (0x00A0)



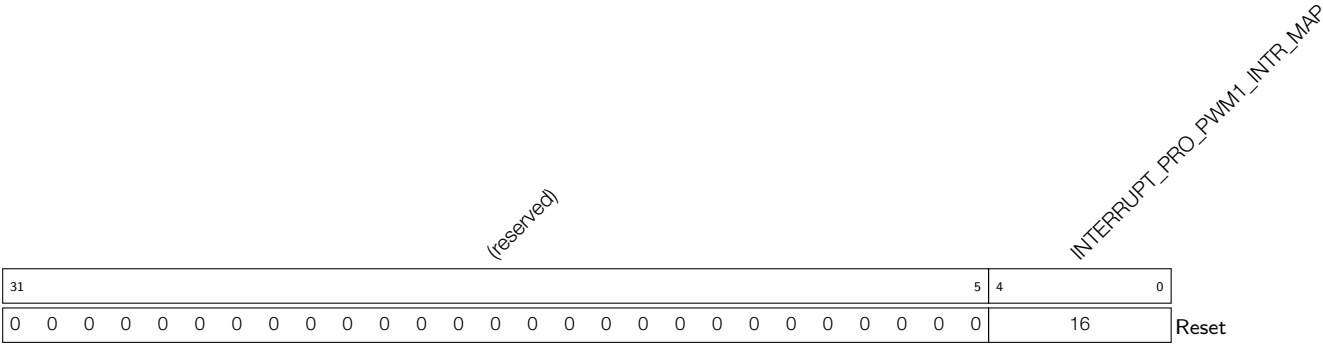
**INTERRUPT\_PRO\_SDIO\_HOST\_INTERRUPT\_MAP** This register is used to map SDIO\_HOST\_INTERRUPT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.42: INTERRUPT\_PRO\_PWM0\_INTR\_MAP\_REG (0x00A4)



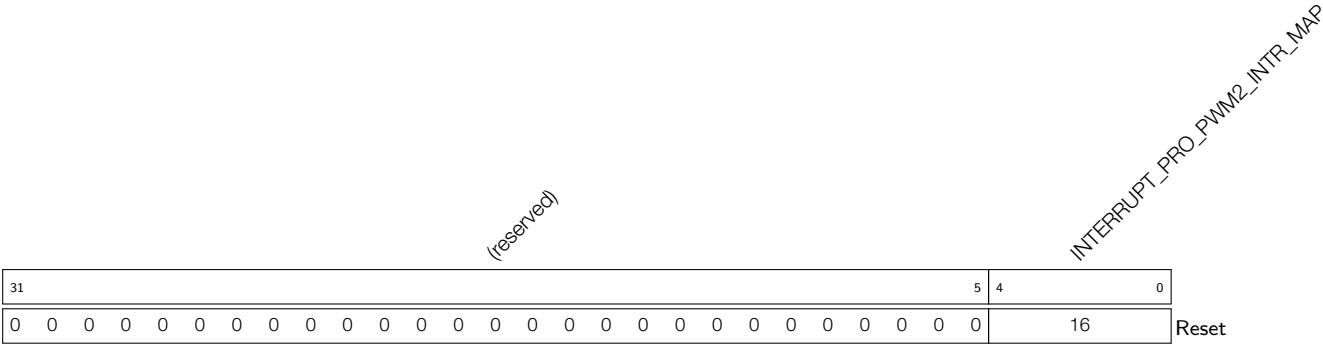
**INTERRUPT\_PRO\_PWM0\_INTR\_MAP** This register is used to map PWM0\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.43: INTERRUPT\_PRO\_PWM1\_INTR\_MAP\_REG (0x00A8)



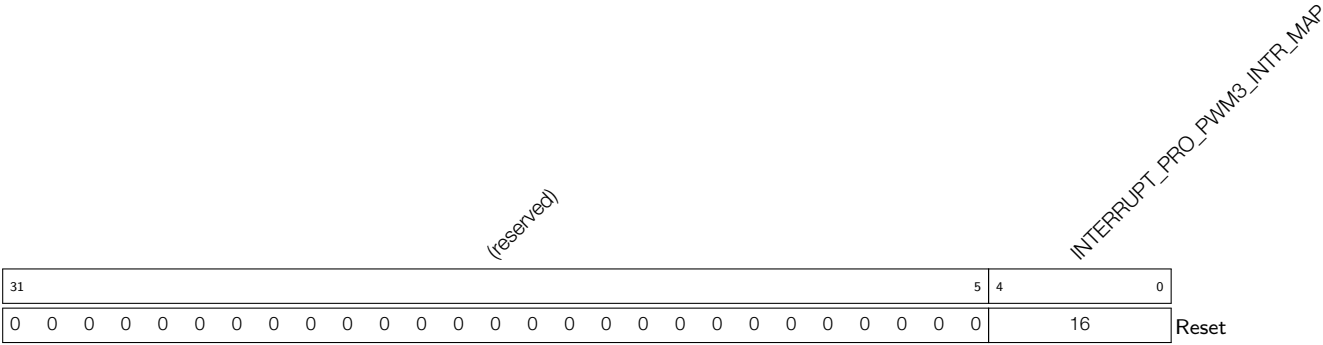
**INTERRUPT\_PRO\_PWM1\_INTR\_MAP** This register is used to map PWM1\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.44: INTERRUPT\_PRO\_PWM2\_INTR\_MAP\_REG (0x00AC)



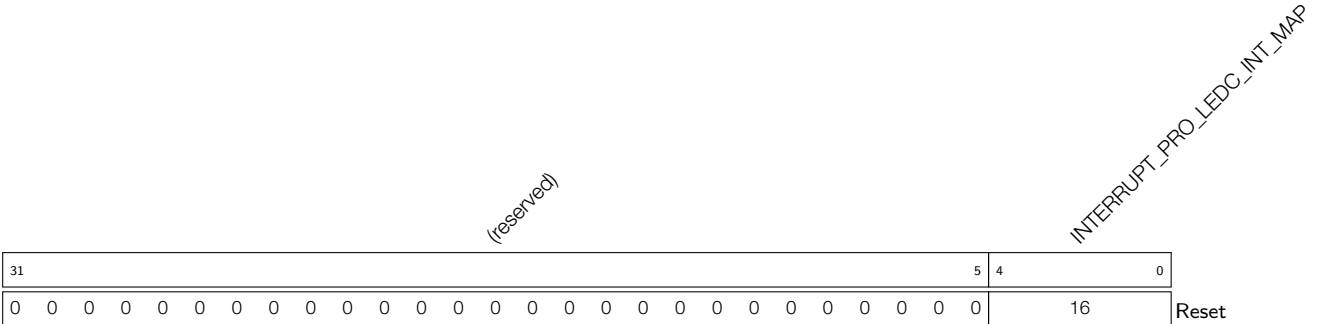
**INTERRUPT\_PRO\_PWM2\_INTR\_MAP** This register is used to map PWM2\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.45: INTERRUPT\_PRO\_PWM3\_INTR\_MAP\_REG (0x00B0)



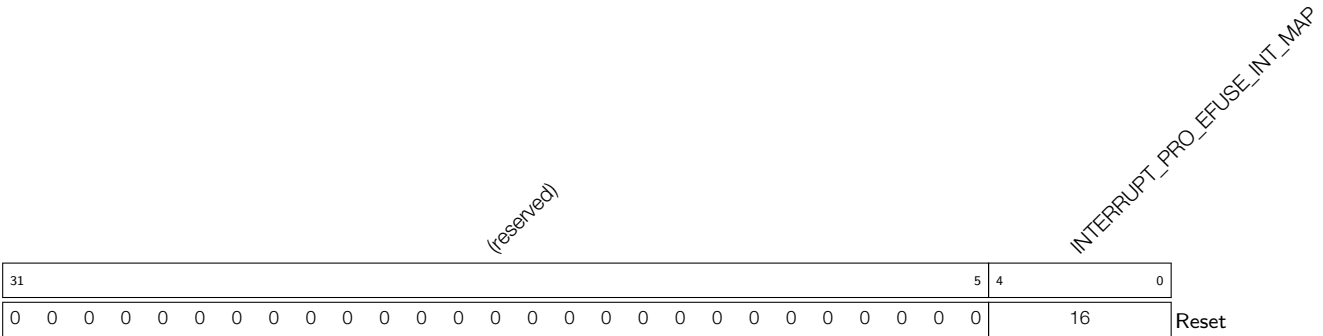
**INTERRUPT\_PRO\_PWM3\_INTR\_MAP** This register is used to map PWM3\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.46: INTERRUPT\_PRO\_LEDC\_INT\_MAP\_REG (0x00B4)



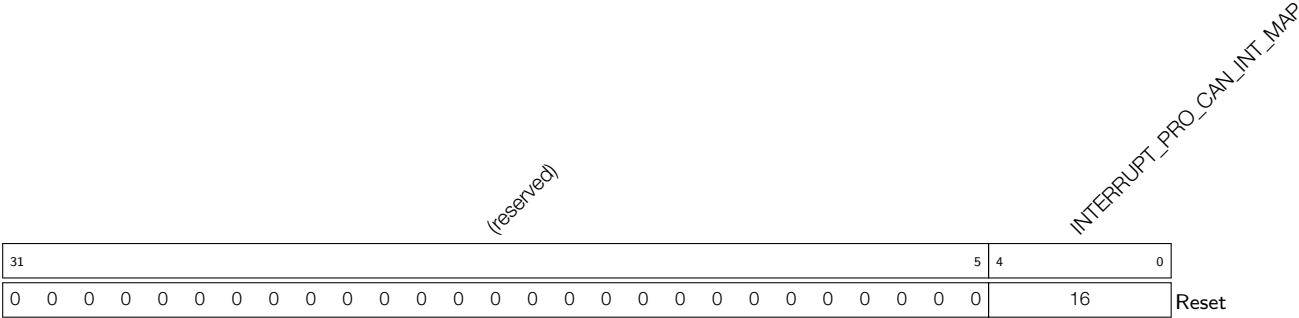
**INTERRUPT\_PRO\_LEDC\_INT\_MAP** This register is used to map LEDC\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.47: INTERRUPT\_PRO\_EFUSE\_INT\_MAP\_REG (0x00B8)



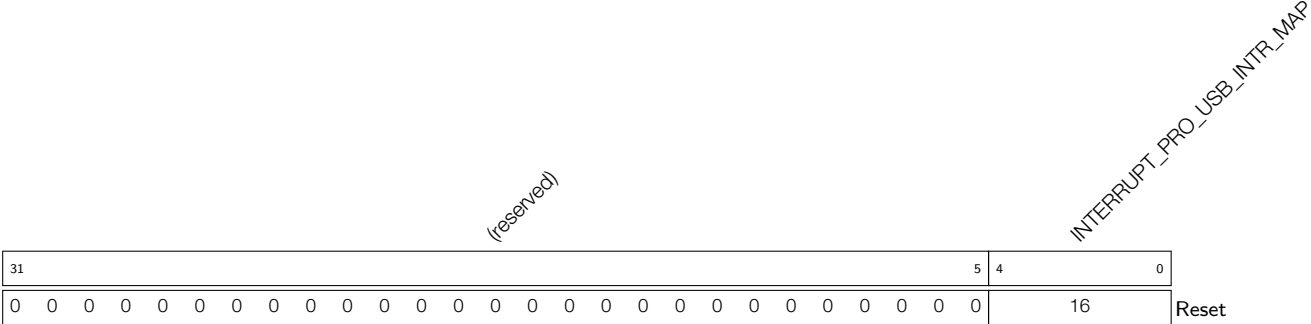
**INTERRUPT\_PRO\_EFUSE\_INT\_MAP** This register is used to map EFUSE\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.48: INTERRUPT\_PRO\_CAN\_INT\_MAP\_REG (0x00BC)



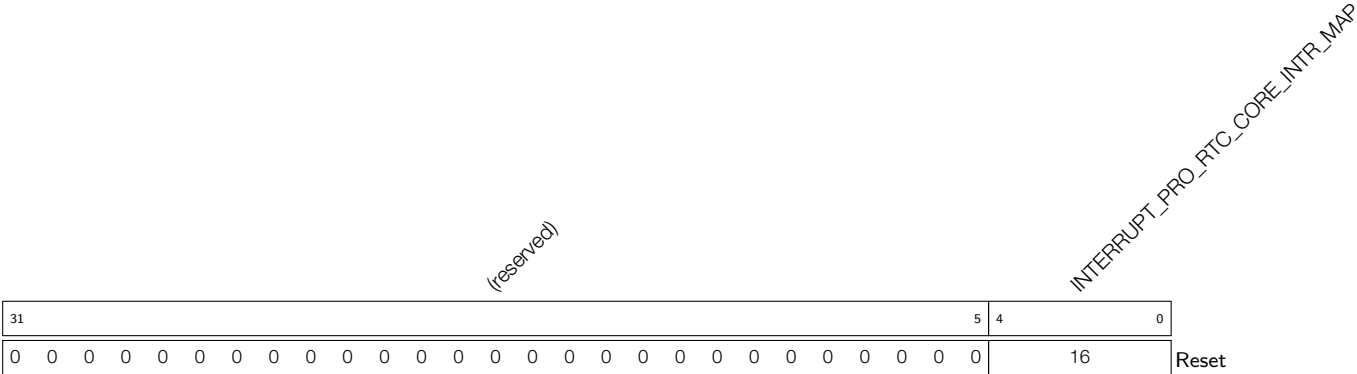
**INTERRUPT\_PRO\_CAN\_INT\_MAP** This register is used to map CAN\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.49: INTERRUPT\_PRO\_USB\_INTR\_MAP\_REG (0x00C0)



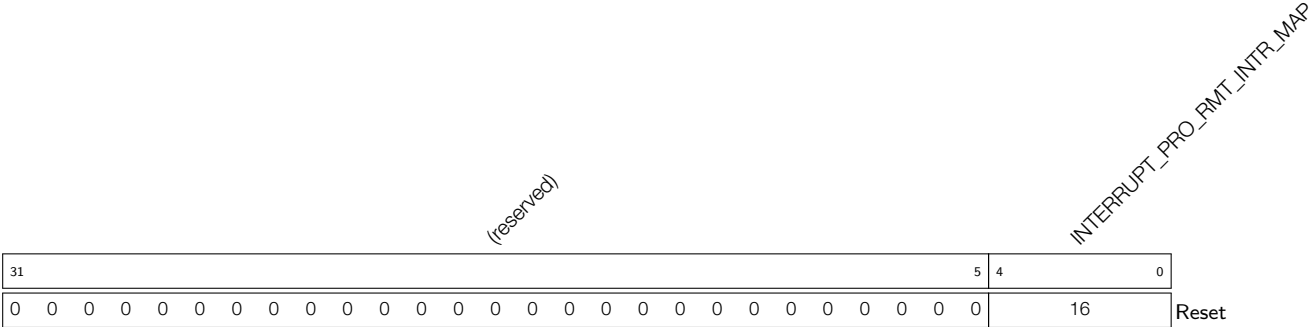
**INTERRUPT\_PRO\_USB\_INTR\_MAP** This register is used to map USB\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.50: INTERRUPT\_PRO\_RTC\_CORE\_INTR\_MAP\_REG (0x00C4)



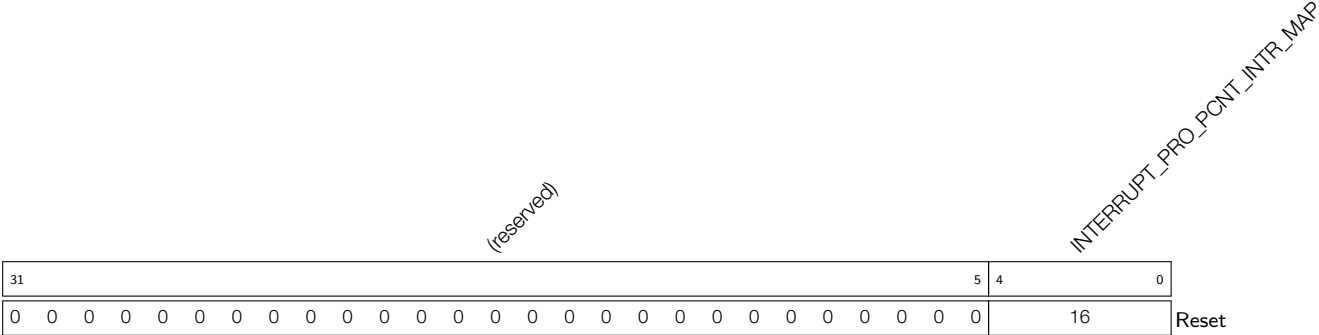
**INTERRUPT\_PRO\_RTC\_CORE\_INTR\_MAP** This register is used to map RTC\_CORE\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.51: INTERRUPT\_PRO\_RMT\_INTR\_MAP\_REG (0x00C8)



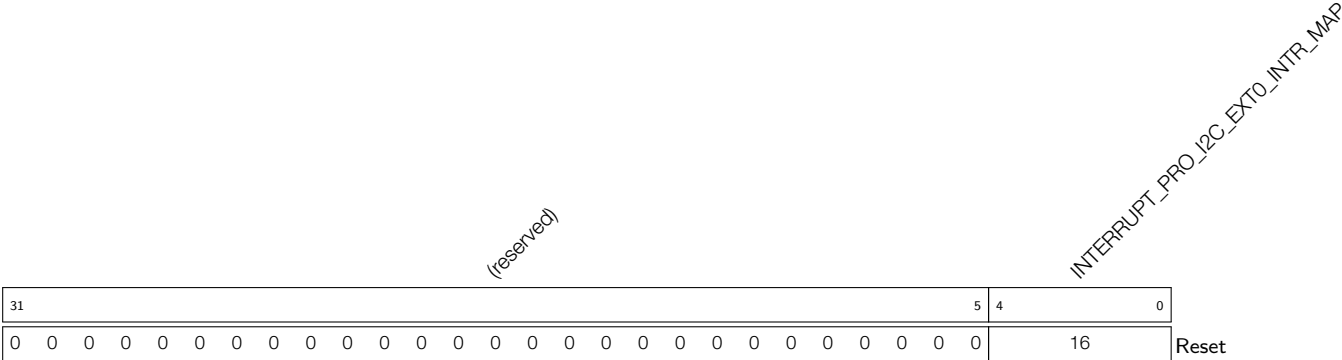
**INTERRUPT\_PRO\_RMT\_INTR\_MAP** This register is used to map RMT\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.52: INTERRUPT\_PRO\_PCNT\_INTR\_MAP\_REG (0x00CC)

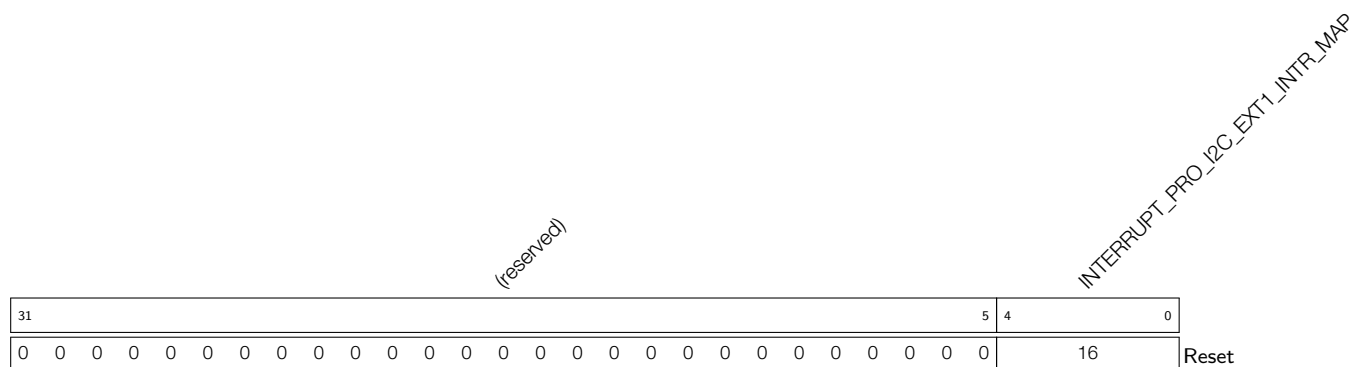


**INTERRUPT\_PRO\_PCNT\_INTR\_MAP** This register is used to map PCNT\_INTR interrupt signal to one of the CPU interrupts. (R/W)

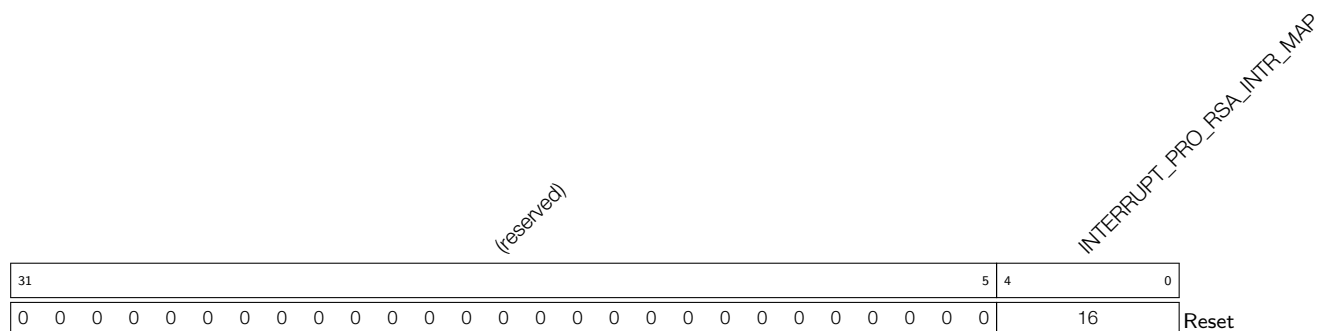
Register 4.53: INTERRUPT\_PRO\_I2C\_EXT0\_INTR\_MAP\_REG (0x00D0)



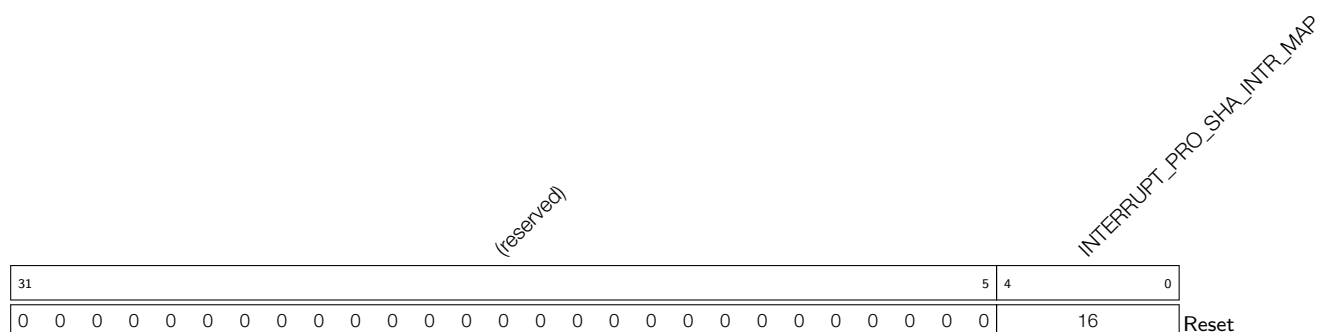
**INTERRUPT\_PRO\_I2C\_EXT0\_INTR\_MAP** This register is used to map I2C\_EXT0\_INTR interrupt signal to one of the CPU interrupts. (R/W)

**Register 4.54: INTERRUPT\_PRO\_I2C\_EXT1\_INTR\_MAP\_REG (0x00D4)**

**INTERRUPT\_PRO\_I2C\_EXT1\_INTR\_MAP** This register is used to map I2C\_EXT1\_INTR interrupt signal to one of the CPU interrupts. (R/W)

**Register 4.55: INTERRUPT\_PRO\_RSA\_INTR\_MAP\_REG (0x00D8)**

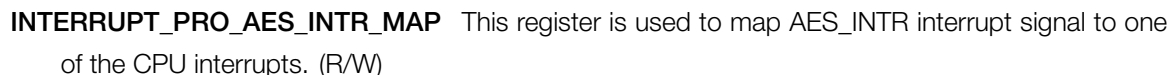
**INTERRUPT\_PRO\_RSA\_INTR\_MAP** This register is used to map RSA\_INTR interrupt signal to one of the CPU interrupts. (R/W)

**Register 4.56: INTERRUPT\_PRO\_SHA\_INTR\_MAP\_REG (0x00DC)**

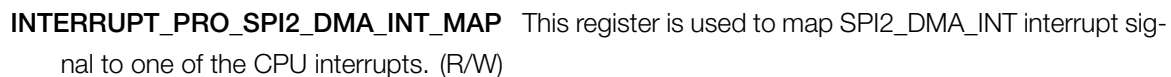
**INTERRUPT\_PRO\_SHA\_INTR\_MAP** This register is used to map SHA\_INTR interrupt signal to one of the CPU interrupts. (R/W)



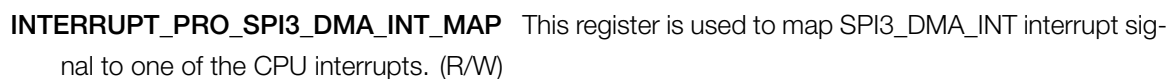
[Submit Documentation Feedback](#)



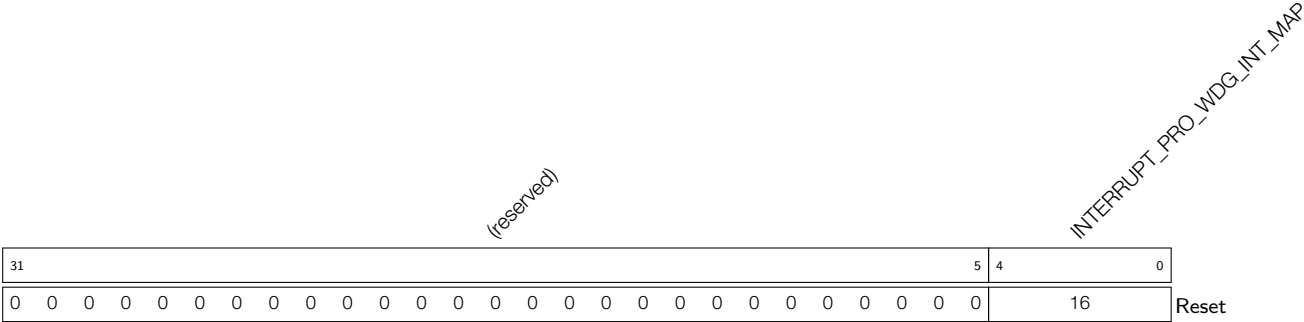
[Submit Documentation Feedback](#)



[Submit Documentation Feedback](#)

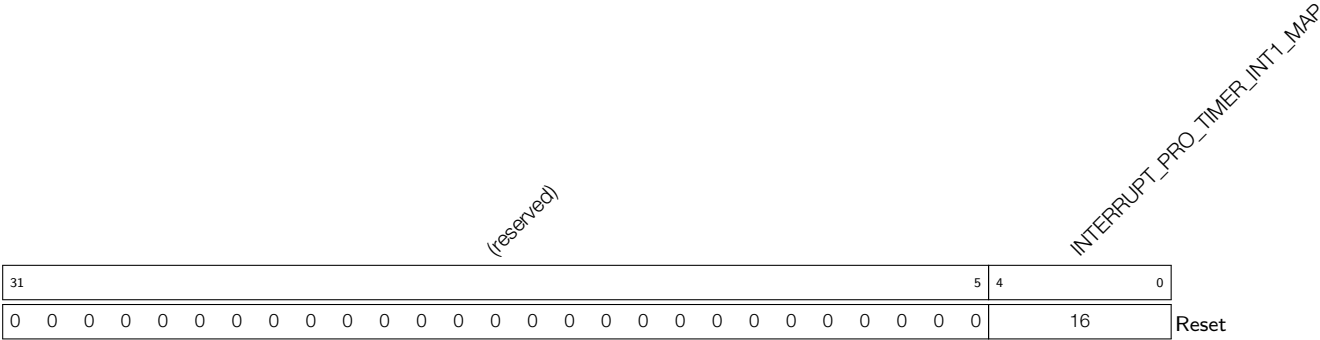


Register 4.60: INTERRUPT\_PRO\_WDG\_INT\_MAP\_REG (0x00EC)



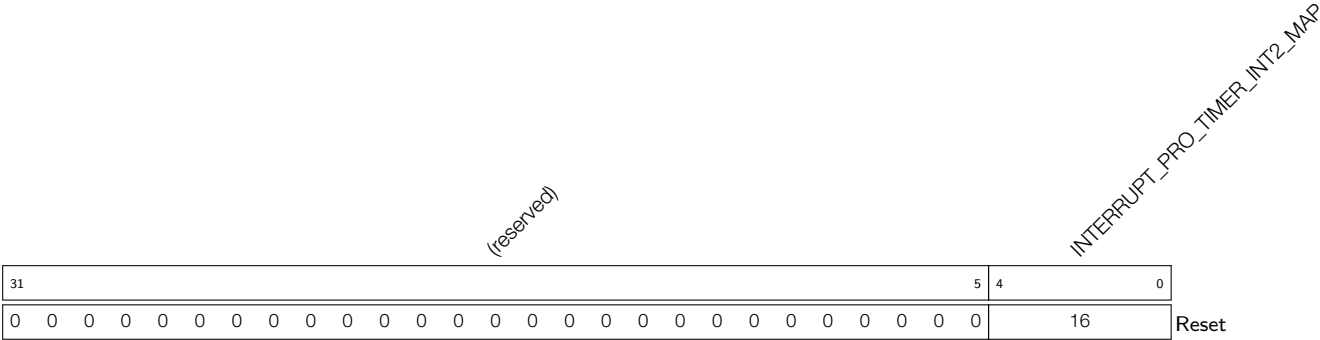
**INTERRUPT\_PRO\_WDG\_INT\_MAP** This register is used to map WDG\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.61: INTERRUPT\_PRO\_TIMER\_INT1\_MAP\_REG (0x00F0)



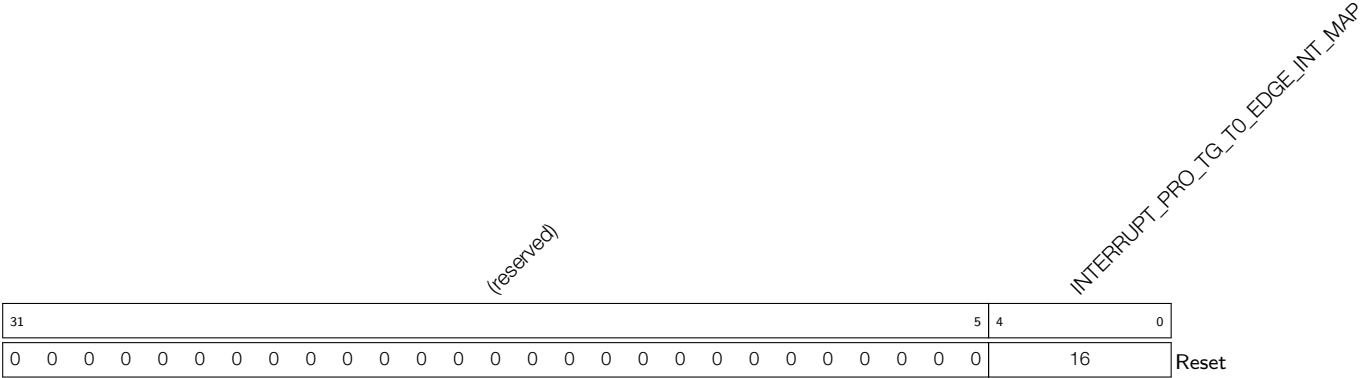
**INTERRUPT\_PRO\_TIMER\_INT1\_MAP** This register is used to map TIMER\_INT1 interrupt signal to one of the CPU interrupts. (R/W)

Register 4.62: INTERRUPT\_PRO\_TIMER\_INT2\_MAP\_REG (0x00F4)



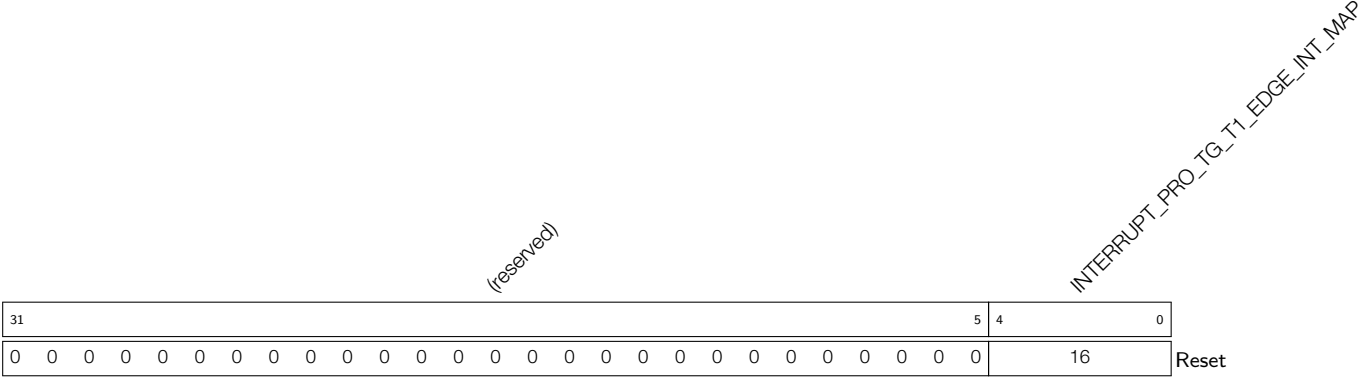
**INTERRUPT\_PRO\_TIMER\_INT2\_MAP** This register is used to map TIMER\_INT2 interrupt signal to one of the CPU interrupts. (R/W)

Register 4.63: INTERRUPT\_PRO\_TG\_T0\_EDGE\_INT\_MAP\_REG (0x00F8)



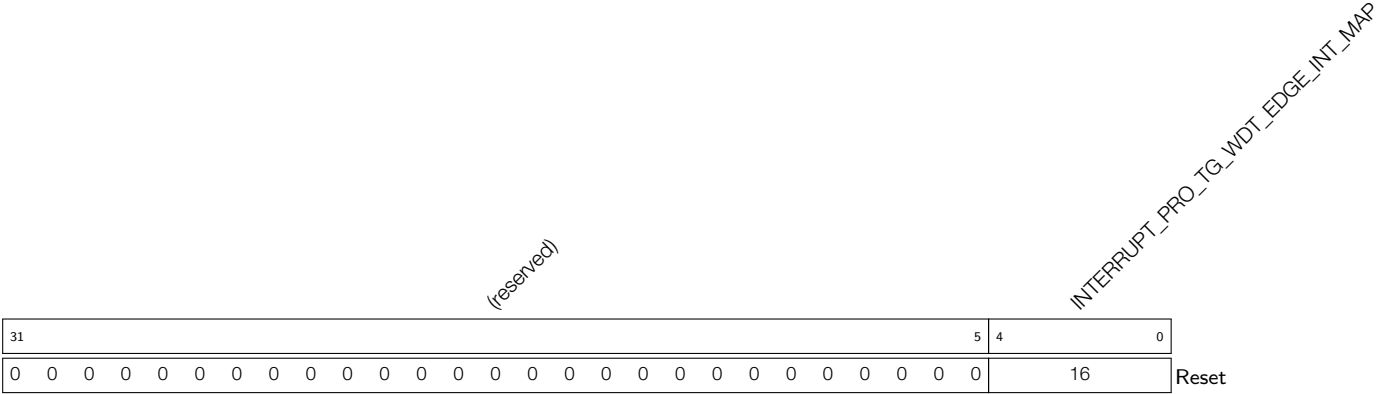
**INTERRUPT\_PRO\_TG\_T0\_EDGE\_INT\_MAP** This register is used to map TG\_T0\_EDGE\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.64: INTERRUPT\_PRO\_TG\_T1\_EDGE\_INT\_MAP\_REG (0x00FC)



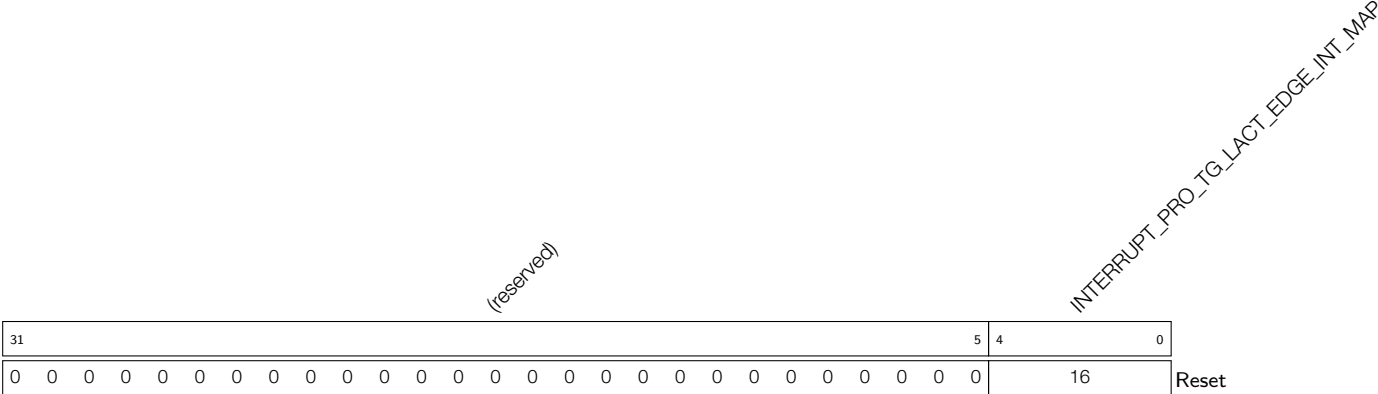
**INTERRUPT\_PRO\_TG\_T1\_EDGE\_INT\_MAP** This register is used to map TG\_T1\_EDGE\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.65: INTERRUPT\_PRO\_TG\_WDT\_EDGE\_INT\_MAP\_REG (0x0100)



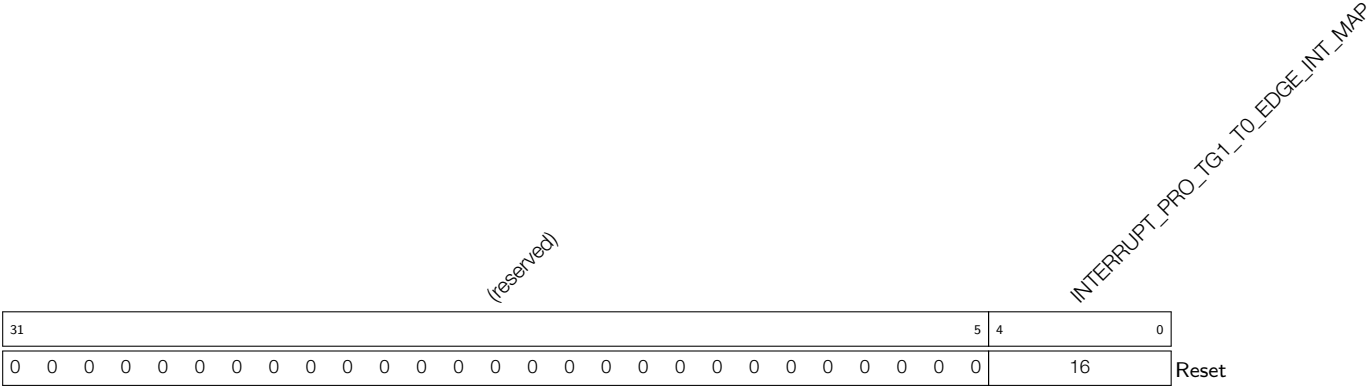
**INTERRUPT\_PRO\_TG\_WDT\_EDGE\_INT\_MAP** This register is used to map TG\_WDT\_EDGE\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.66: INTERRUPT\_PRO\_TG\_LACT\_EDGE\_INT\_MAP\_REG (0x0104)



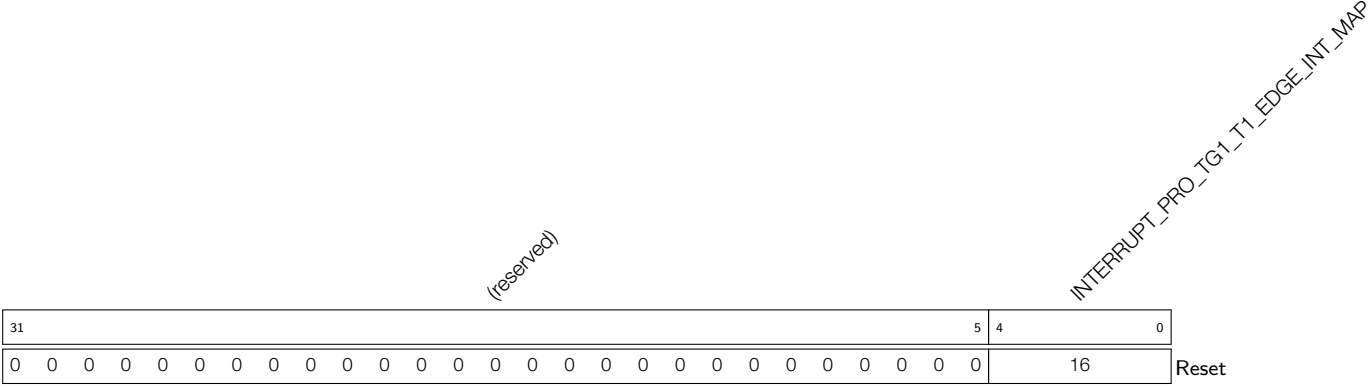
**INTERRUPT\_PRO\_TG\_LACT\_EDGE\_INT\_MAP** This register is used to map TG\_LACT\_EDGE\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.67: INTERRUPT\_PRO\_TG1\_T0\_EDGE\_INT\_MAP\_REG (0x0108)



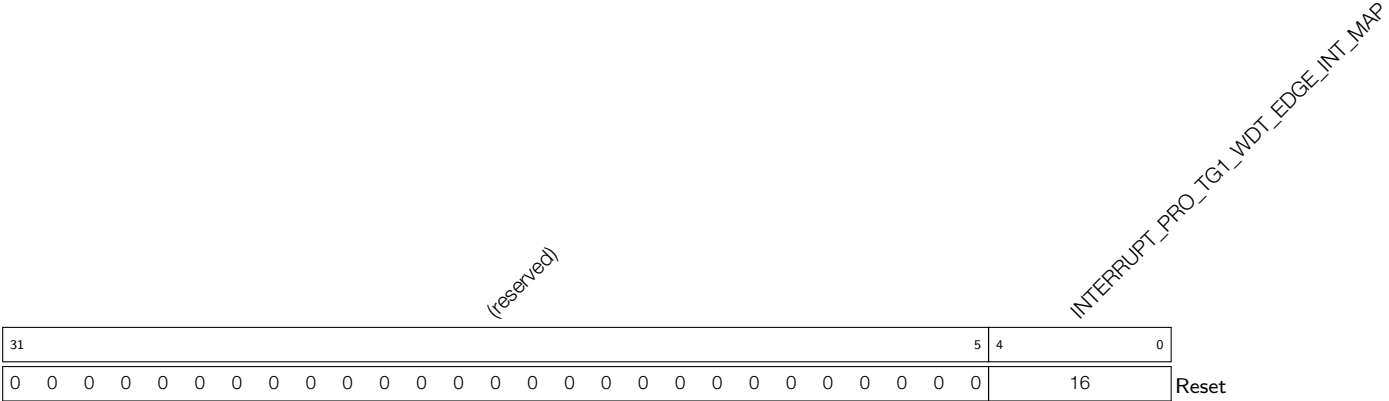
**INTERRUPT\_PRO\_TG1\_T0\_EDGE\_INT\_MAP** This register is used to map TG1\_T0\_EDGE\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.68: INTERRUPT\_PRO\_TG1\_T1\_EDGE\_INT\_MAP\_REG (0x010C)



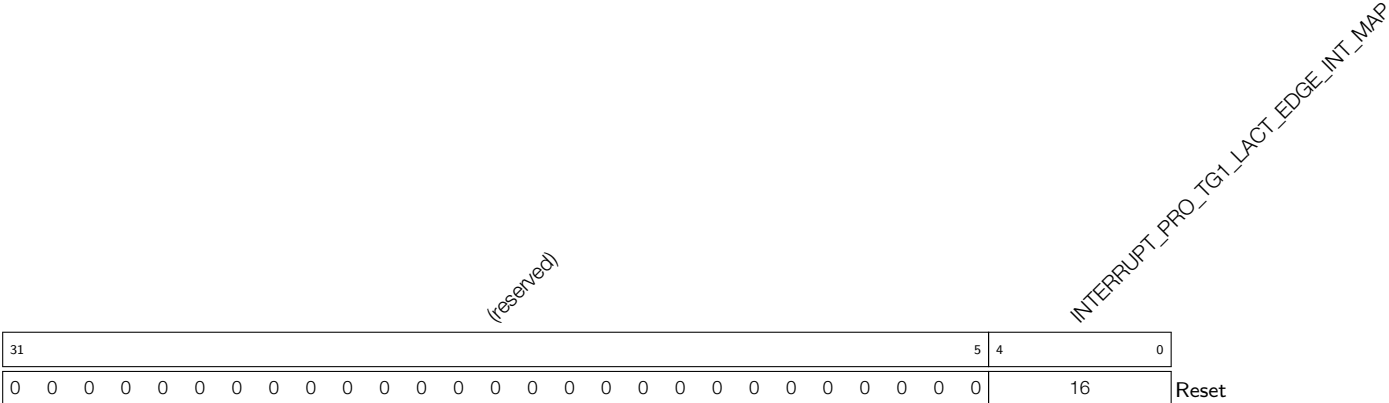
**INTERRUPT\_PRO\_TG1\_T1\_EDGE\_INT\_MAP** This register is used to map TG1\_T1\_EDGE\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.69: INTERRUPT\_PRO\_TG1\_WDT\_EDGE\_INT\_MAP\_REG (0x0110)

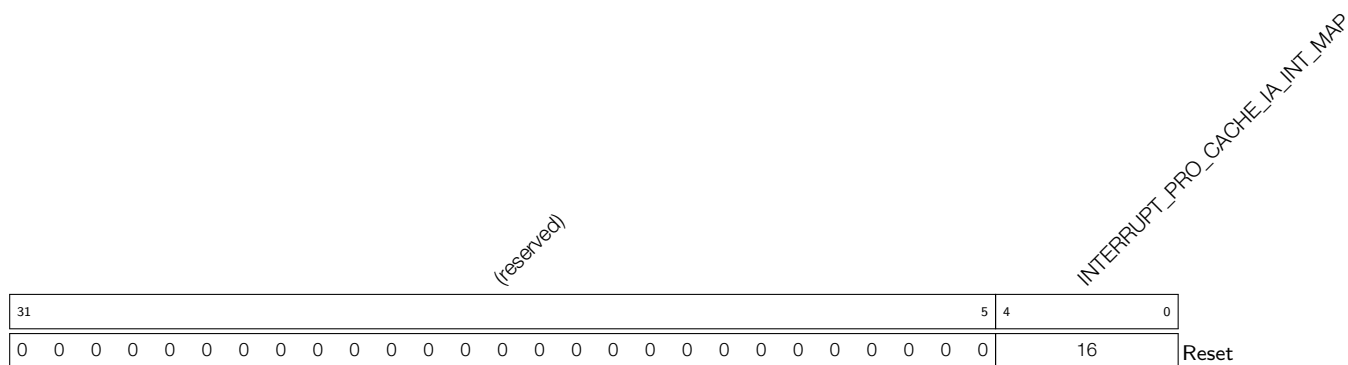


**INTERRUPT\_PRO\_TG1\_WDT\_EDGE\_INT\_MAP** This register is used to map TG1\_WDT\_EDGE\_INT interrupt signal to one of the CPU interrupts. (R/W)

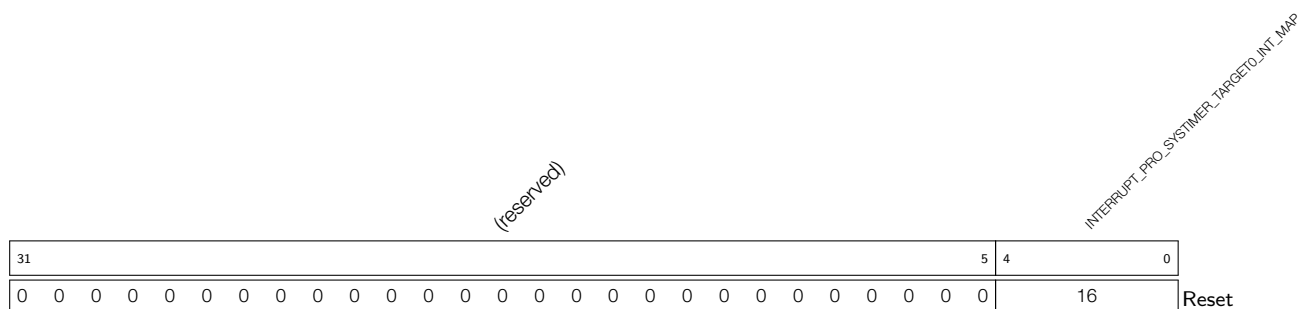
Register 4.70: INTERRUPT\_PRO\_TG1\_LACT\_EDGE\_INT\_MAP\_REG (0x0114)



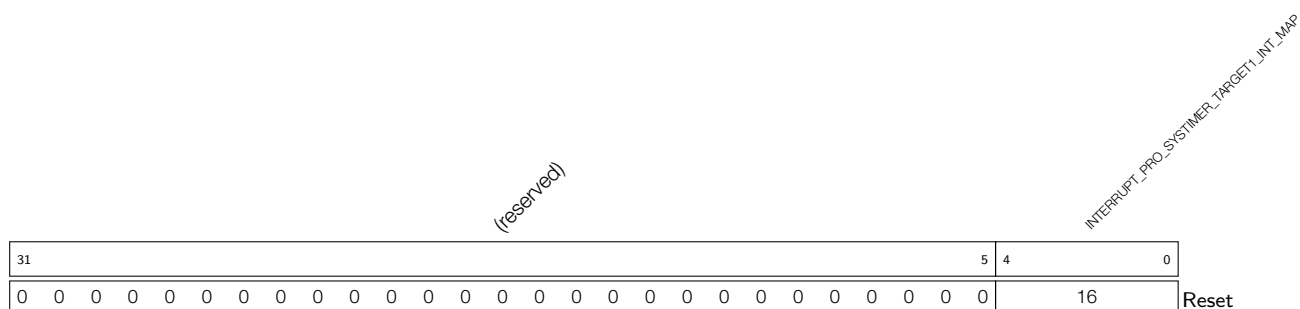
**INTERRUPT\_PRO\_TG1\_LACT\_EDGE\_INT\_MAP** This register is used to map TG1\_LACT\_EDGE\_INT interrupt signal to one of the CPU interrupts. (R/W)

**Register 4.71: INTERRUPT\_PRO\_CACHE\_IA\_INT\_MAP\_REG (0x0118)**

**INTERRUPT\_PRO\_CACHE\_IA\_INT\_MAP** This register is used to map CACHE\_IA\_INT interrupt signal to one of the CPU interrupts. (R/W)

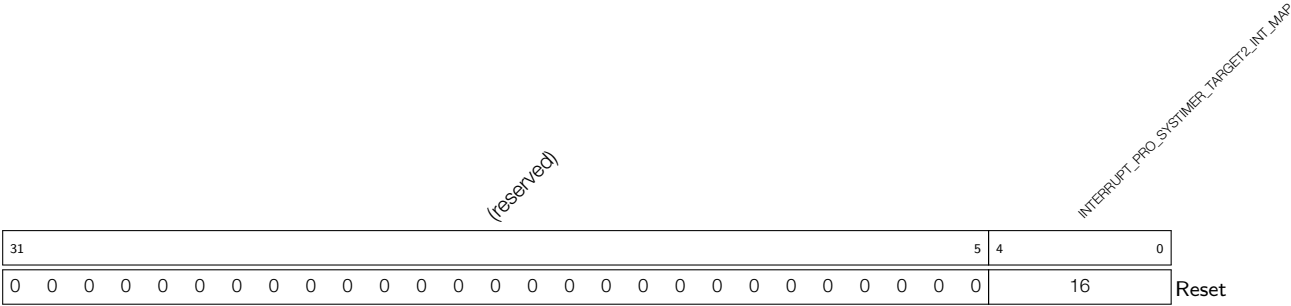
**Register 4.72: INTERRUPT\_PRO\_SYSTIMER\_TARGET0\_INT\_MAP\_REG (0x011C)**

**INTERRUPT\_PRO\_SYSTIMER\_TARGET0\_INT\_MAP** This register is used to map SYSTIMER\_TARGET0\_INT interrupt signal to one of the CPU interrupts. (R/W)

**Register 4.73: INTERRUPT\_PRO\_SYSTIMER\_TARGET1\_INT\_MAP\_REG (0x0120)**

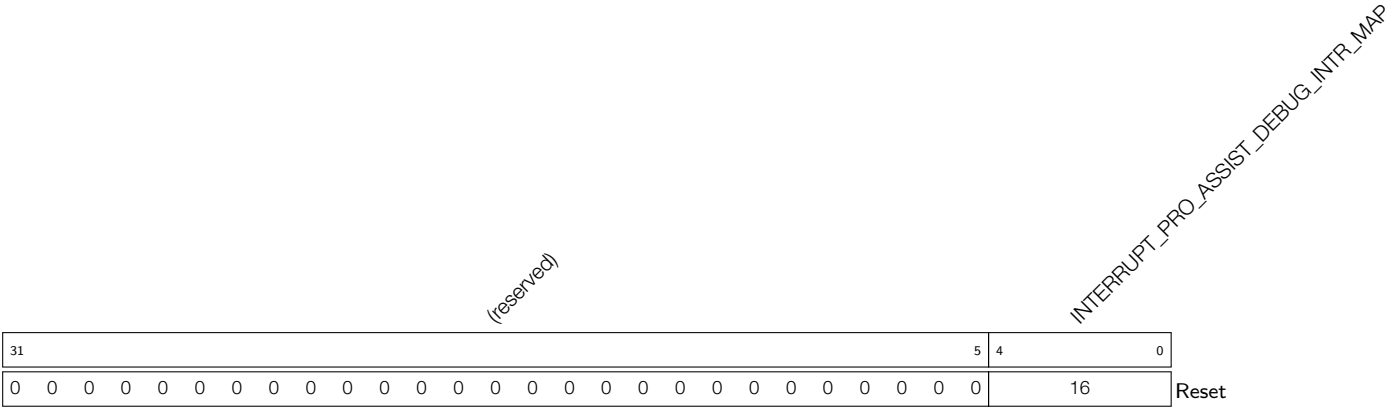
**INTERRUPT\_PRO\_SYSTIMER\_TARGET1\_INT\_MAP** This register is used to map SYSTIMER\_TARGET1\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.74: INTERRUPT\_PRO\_SYSTIMER\_TARGET2\_INT\_MAP\_REG (0x0124)



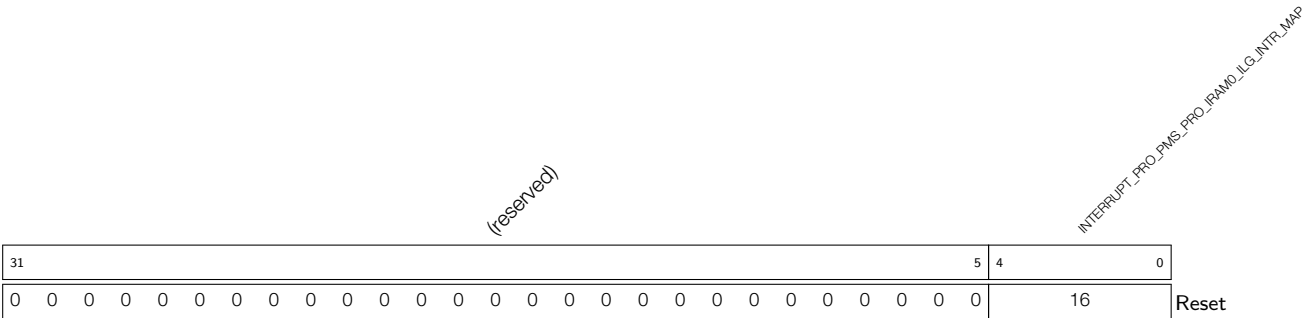
**INTERRUPT\_PRO\_SYSTIMER\_TARGET2\_INT\_MAP** This register is used to map SYSTIMER\_TARGET2\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.75: INTERRUPT\_PRO\_ASSIST\_DEBUG\_INTR\_MAP\_REG (0x0128)



**INTERRUPT\_PRO\_ASSIST\_DEBUG\_INTR\_MAP** This register is used to map ASSIST\_DEBUG\_INTR interrupt signal to one of the CPU interrupts. (R/W)

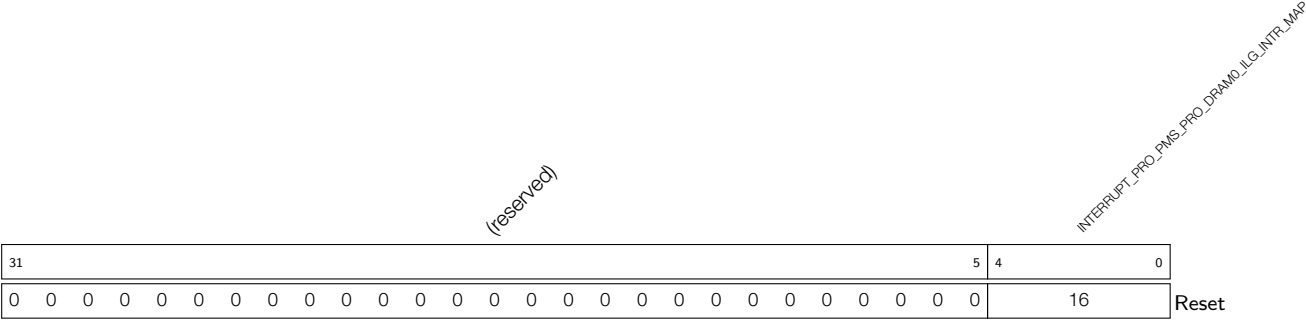
Register 4.76: INTERRUPT\_PRO\_PMS\_PRO\_IRAM0\_ILG\_INTR\_MAP\_REG (0x012C)



**INTERRUPT\_PRO\_PMS\_PRO\_IRAM0\_ILG\_INTR\_MAP** This register is used to map PMS\_PRO\_IRAM0\_ILG\_INTR interrupt signal to one of the CPU interrupts. (R/W)

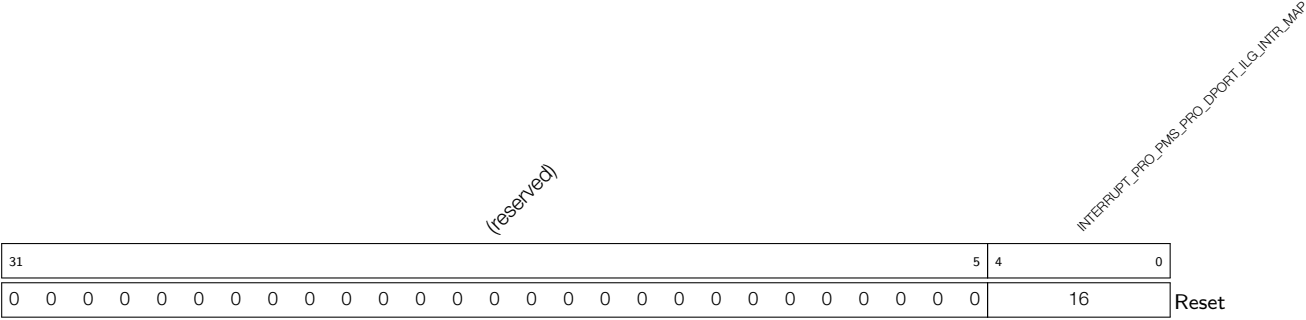


Register 4.77: INTERRUPT\_PRO\_PMS\_PRO\_DRAM0\_ILG\_INTR\_MAP\_REG (0x0130)



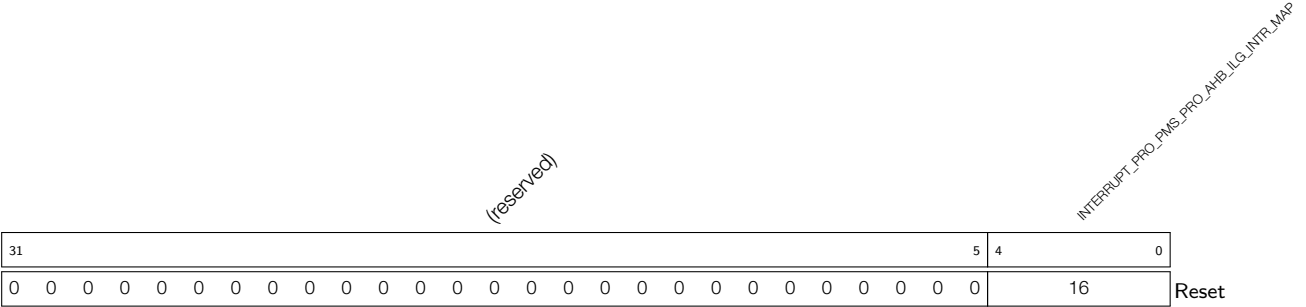
**INTERRUPT\_PRO\_PMS\_PRO\_DRAM0\_ILG\_INTR\_MAP** This register is used to map PMS\_PRO\_DRAM0\_ILG\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.78: INTERRUPT\_PRO\_PMS\_PRO\_DPORT\_ILG\_INTR\_MAP\_REG (0x0134)



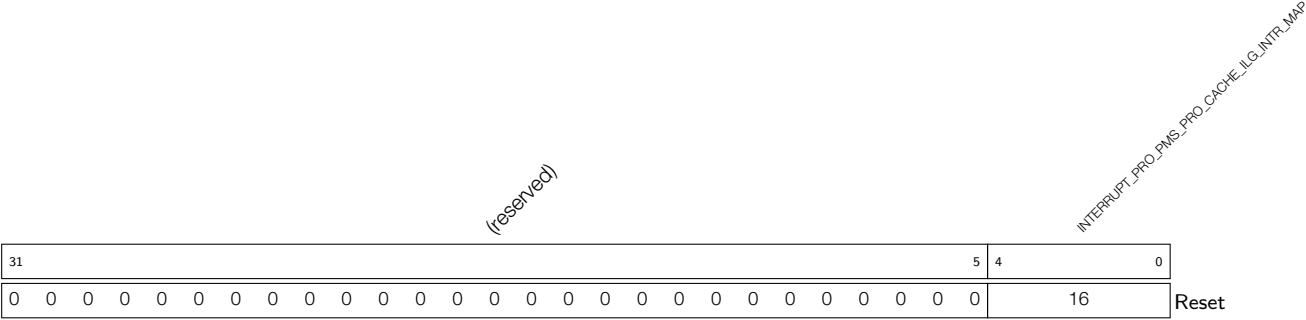
**INTERRUPT\_PRO\_PMS\_PRO\_DPORT\_ILG\_INTR\_MAP** This register is used to map PMS\_PRO\_DPORT\_ILG\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.79: INTERRUPT\_PRO\_PMS\_PRO\_AHB\_ILG\_INTR\_MAP\_REG (0x0138)



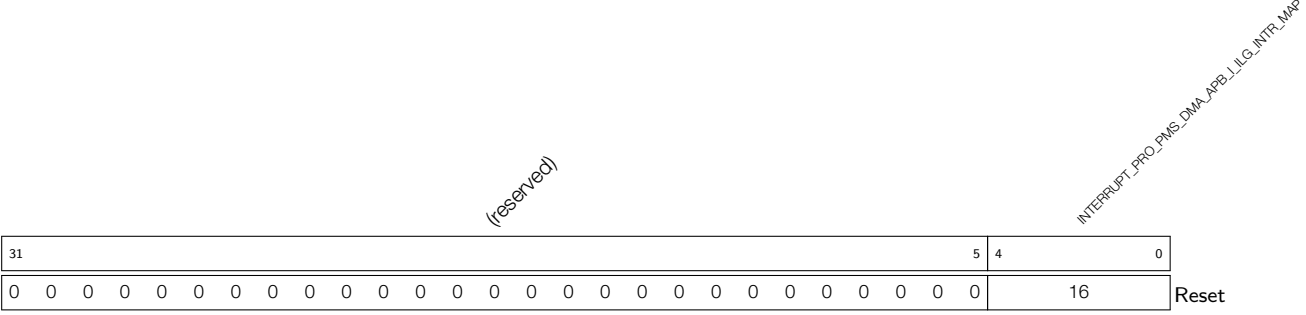
**INTERRUPT\_PRO\_PMS\_PRO\_AHB\_ILG\_INTR\_MAP** This register is used to map PMS\_PRO\_AHB\_ILG\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.80: INTERRUPT\_PRO\_PMS\_PRO\_CACHE\_ILG\_INTR\_MAP\_REG (0x013C)



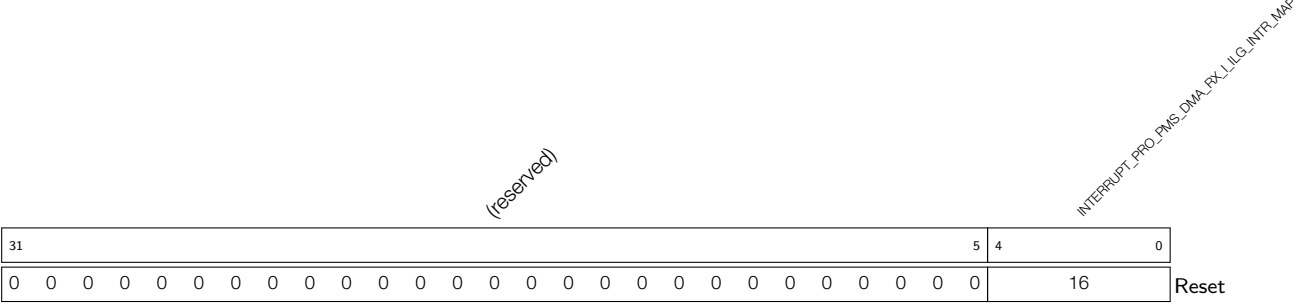
**INTERRUPT\_PRO\_PMS\_PRO\_CACHE\_ILG\_INTR\_MAP** This register is used to map PMS\_PRO\_CACHE\_ILG\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.81: INTERRUPT\_PRO\_PMS\_DMA\_APB\_I\_ILG\_INTR\_MAP\_REG (0x0140)



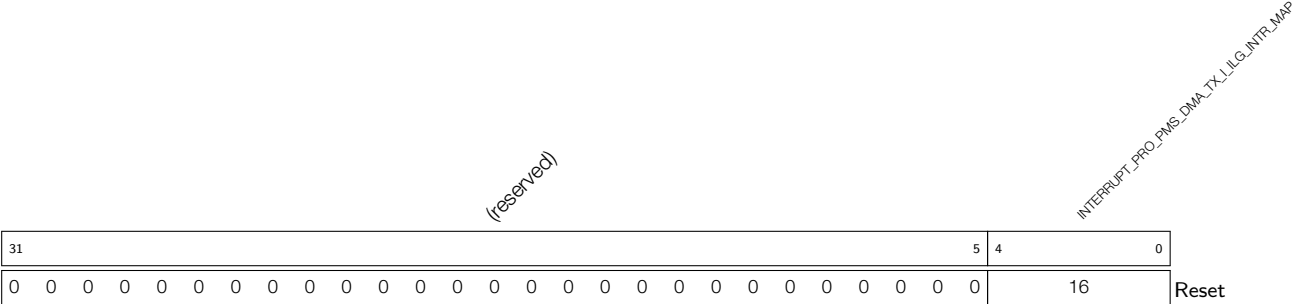
**INTERRUPT\_PRO\_PMS\_DMA\_APB\_I\_ILG\_INTR\_MAP** This register is used to map PMS\_DMA\_APB\_I\_ILG\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.82: INTERRUPT\_PRO\_PMS\_DMA\_RX\_I\_ILG\_INTR\_MAP\_REG (0x0144)



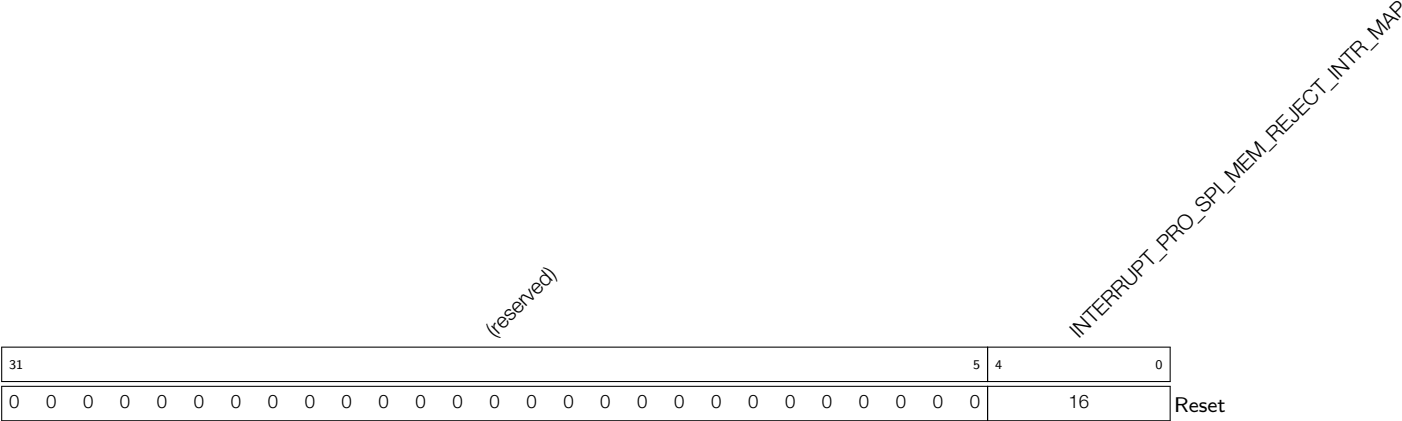
**INTERRUPT\_PRO\_PMS\_DMA\_RX\_I\_ILG\_INTR\_MAP** This register is used to map PMS\_DMA\_RX\_I\_ILG\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.83: INTERRUPT\_PRO\_PMS\_DMA\_TX\_I\_ILG\_INTR\_MAP\_REG (0x0148)



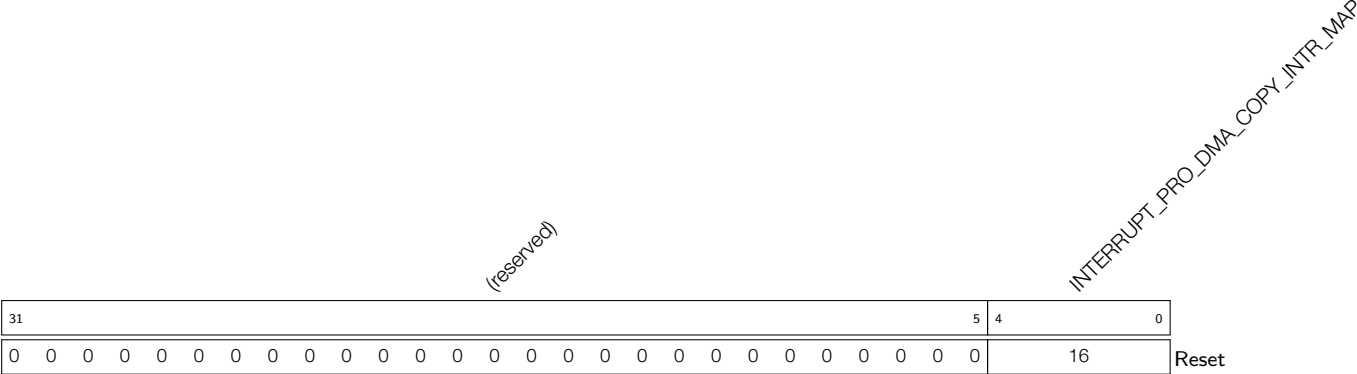
**INTERRUPT\_PRO\_PMS\_DMA\_TX\_I\_ILG\_INTR\_MAP** This register is used to map PMS\_DMA\_TX\_I\_ILG\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.84: INTERRUPT\_PRO\_SPI\_MEM\_REJECT\_INTR\_MAP\_REG (0x014C)



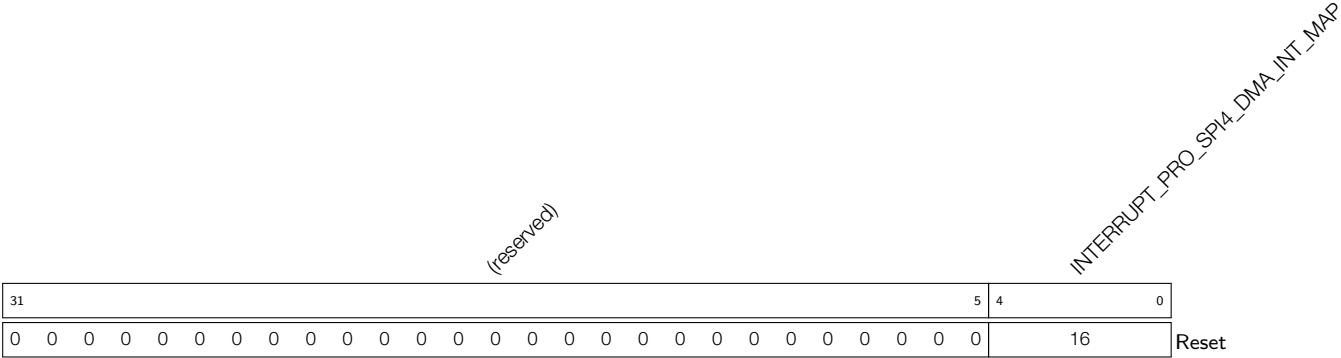
**INTERRUPT\_PRO\_SPI\_MEM\_REJECT\_INTR\_MAP** This register is used to map SPI\_MEM\_REJECT\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.85: INTERRUPT\_PRO\_DMA\_COPY\_INTR\_MAP\_REG (0x0150)



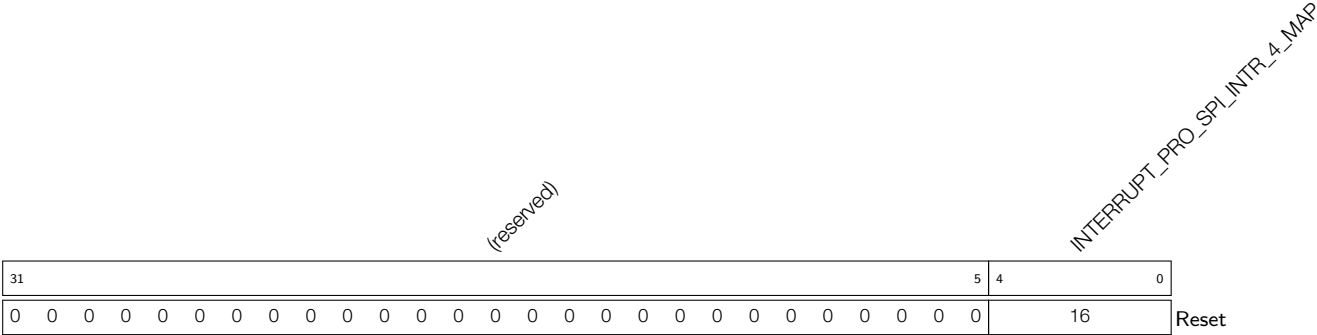
**INTERRUPT\_PRO\_DMA\_COPY\_INTR\_MAP** This register is used to map DMA\_COPY\_INTR interrupt signal to one of the CPU interrupts. (R/W)

Register 4.86: INTERRUPT\_PRO\_SPI4\_DMA\_INT\_MAP\_REG (0x0154)



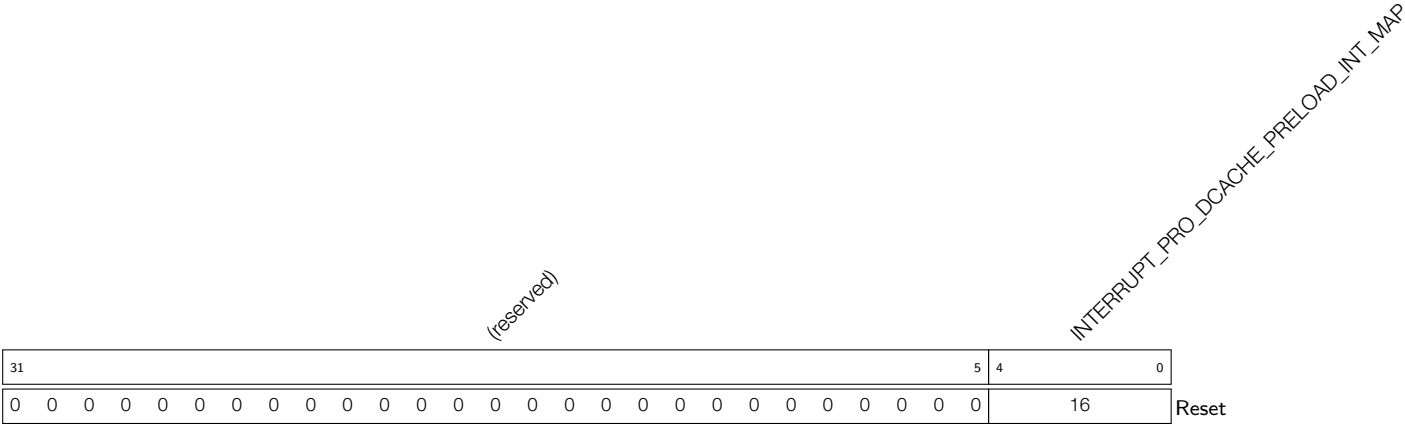
**INTERRUPT\_PRO\_SPI4\_DMA\_INT\_MAP** This register is used to map SPI4\_DMA\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.87: INTERRUPT\_PRO\_SPI\_INTR\_4\_MAP\_REG (0x0158)



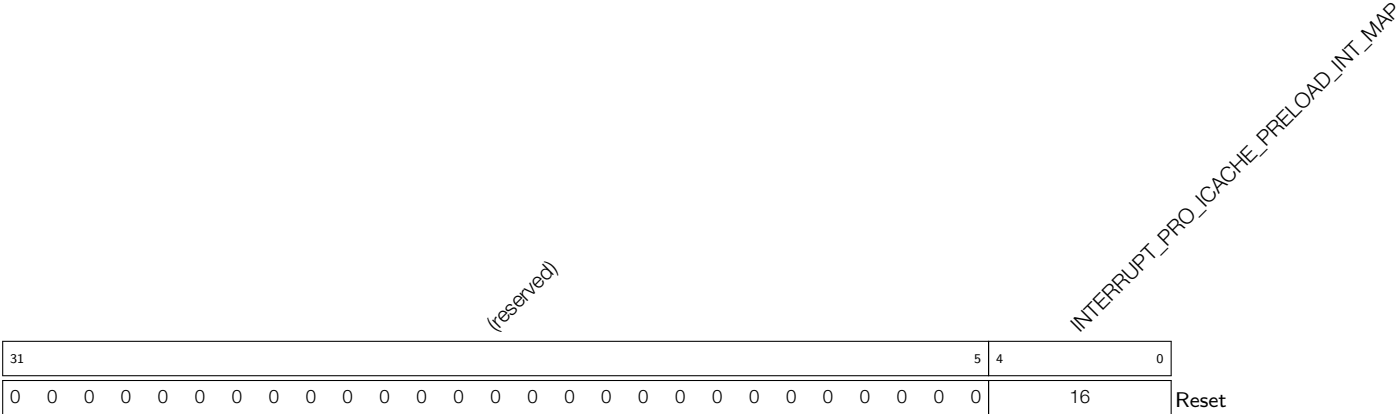
**INTERRUPT\_PRO\_SPI\_INTR\_4\_MAP** This register is used to map SPI\_INTR\_4 interrupt signal to one of the CPU interrupts. (R/W)

Register 4.88: INTERRUPT\_PRO\_DCACHE\_PRELOAD\_INT\_MAP\_REG (0x015C)



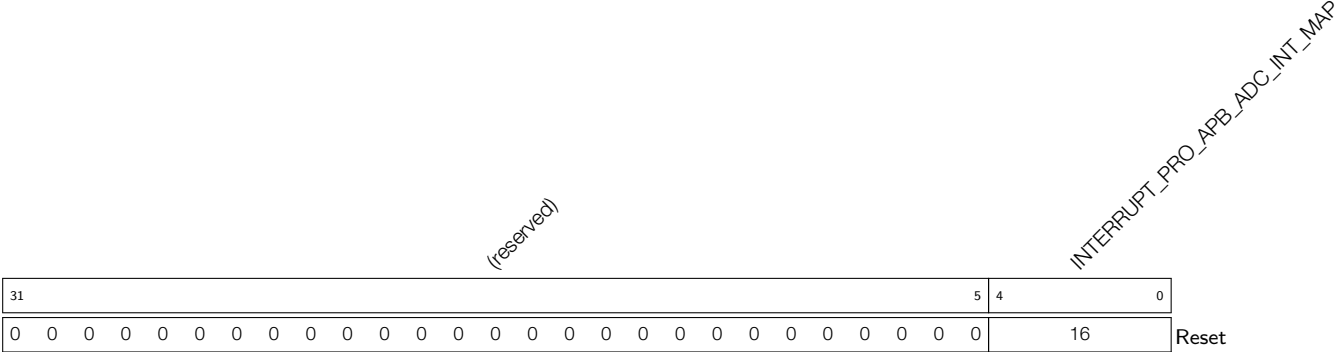
**INTERRUPT\_PRO\_DCACHE\_PRELOAD\_INT\_MAP** This register is used to map DCACHE\_PRELOAD\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.89: INTERRUPT\_PRO\_ICACHE\_PRELOAD\_INT\_MAP\_REG (0x0160)



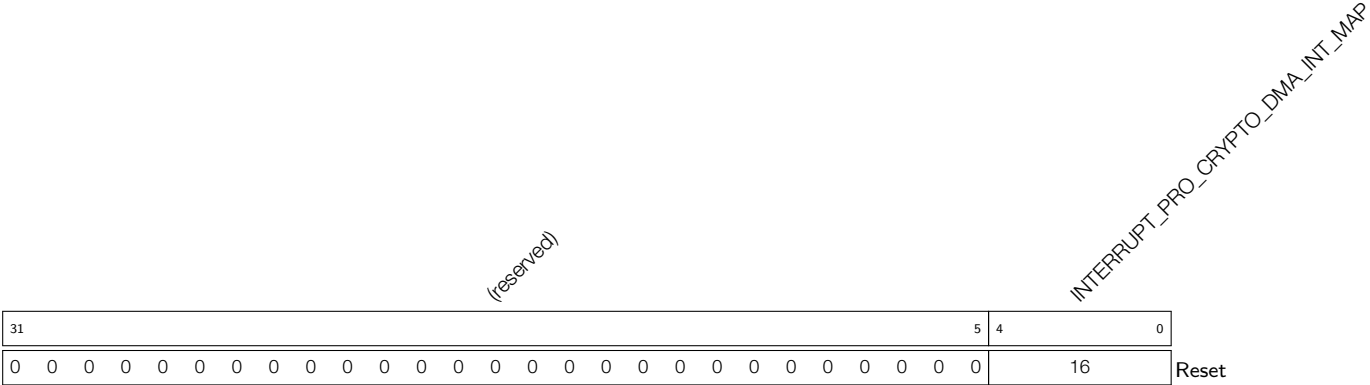
**INTERRUPT\_PRO\_ICACHE\_PRELOAD\_INT\_MAP** This register is used to map ICACHE\_PRELOAD\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.90: INTERRUPT\_PRO\_APB\_ADC\_INT\_MAP\_REG (0x0164)



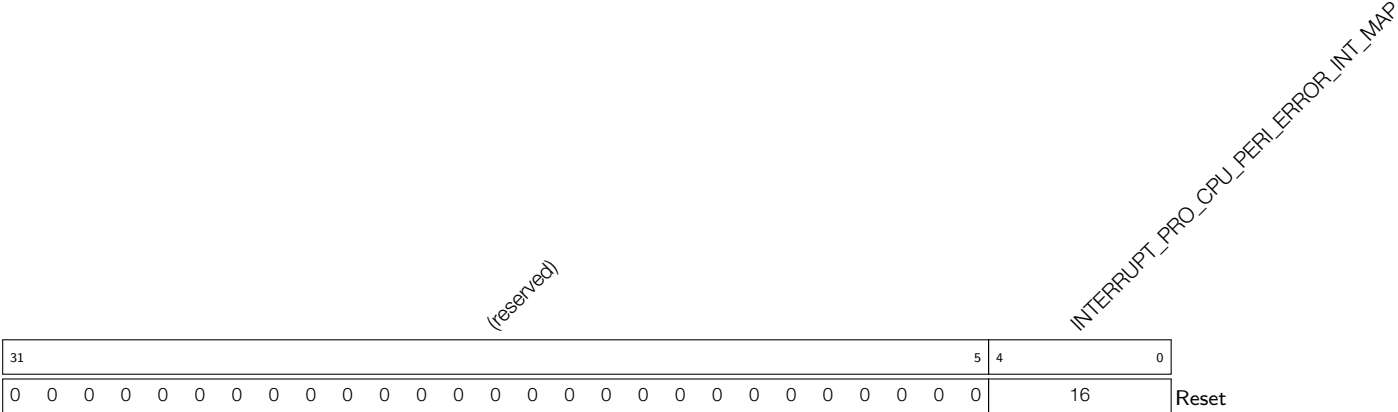
**INTERRUPT\_PRO\_APB\_ADC\_INT\_MAP** This register is used to map APB\_ADC\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.91: INTERRUPT\_PRO\_CRYPTO\_DMA\_INT\_MAP\_REG (0x0168)



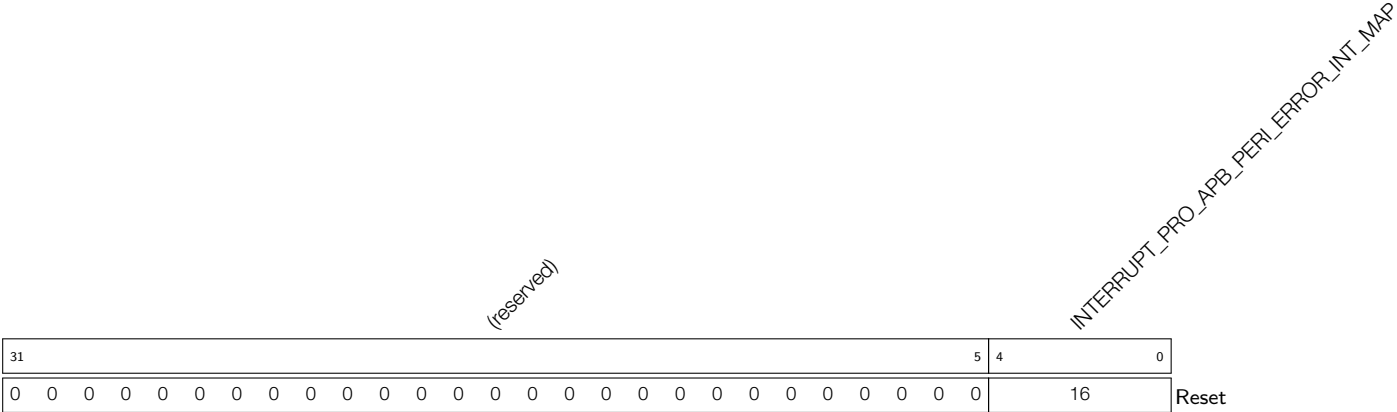
**INTERRUPT\_PRO\_CRYPTO\_DMA\_INT\_MAP** This register is used to map CRYPTO\_DMA\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.92: INTERRUPT\_PRO\_CPU\_PERI\_ERROR\_INT\_MAP\_REG (0x016C)



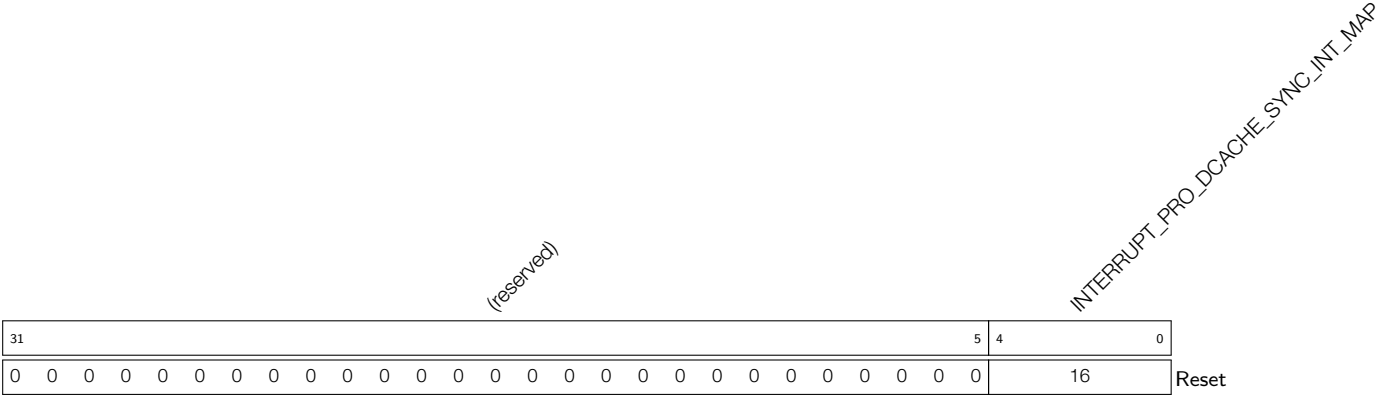
**INTERRUPT\_PRO\_CPU\_PERI\_ERROR\_INT\_MAP** This register is used to map CPU\_PERI\_ERROR\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.93: INTERRUPT\_PRO\_APB\_PERI\_ERROR\_INT\_MAP\_REG (0x0170)



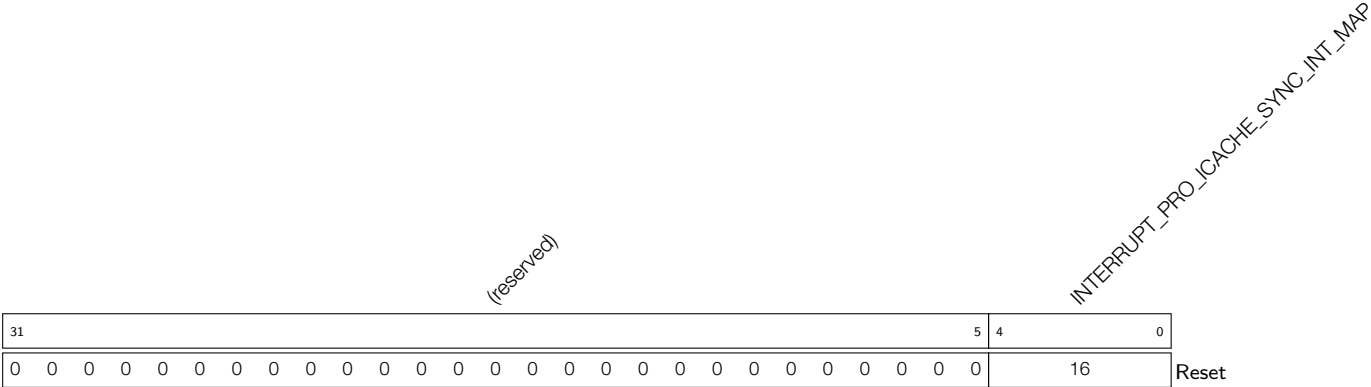
**INTERRUPT\_PRO\_APB\_PERI\_ERROR\_INT\_MAP** This register is used to map APB\_PERI\_ERROR\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.94: INTERRUPT\_PRO\_DCACHE\_SYNC\_INT\_MAP\_REG (0x0174)



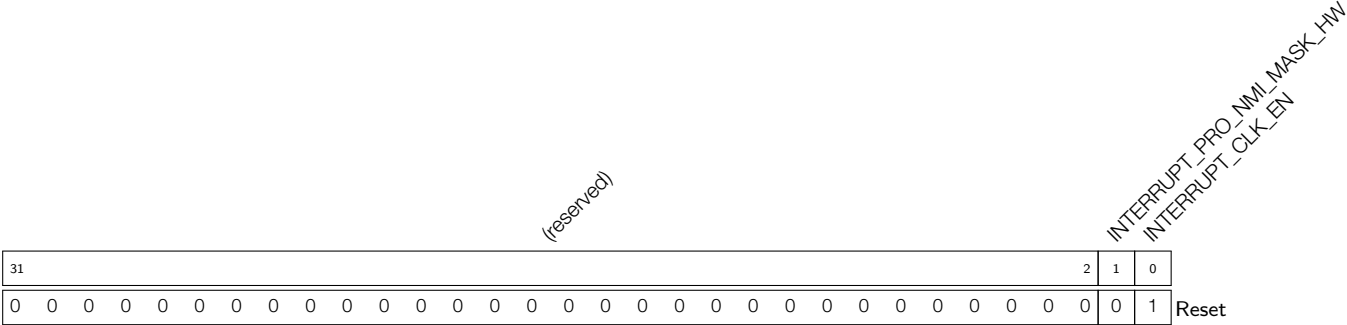
**INTERRUPT\_PRO\_DCACHE\_SYNC\_INT\_MAP** This register is used to map DCACHE\_SYNC\_INT interrupt signal to one of the CPU interrupts. (R/W)

Register 4.95: INTERRUPT\_PRO\_ICACHE\_SYNC\_INT\_MAP\_REG (0x0178)



**INTERRUPT\_PRO\_ICACHE\_SYNC\_INT\_MAP** This register is used to map ICACHE\_SYNC\_INT interrupt signal to one of the CPU interrupts. (R/W)

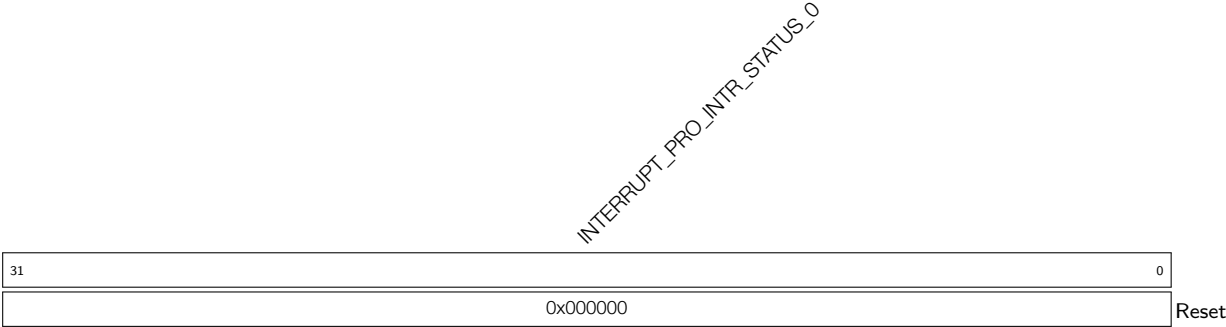
Register 4.96: INTERRUPT\_CLOCK\_GATE\_REG (0x0188)



**INTERRUPT\_CLK\_EN** This bit is used to enable or disable the clock of interrupt matrix. 1: enable the clock; 0: disable the clock. (R/W)

**INTERRUPT\_PRO\_NMI\_MASK\_HW** This bit is used to disable all NMI interrupt signals to CPU. (R/W)

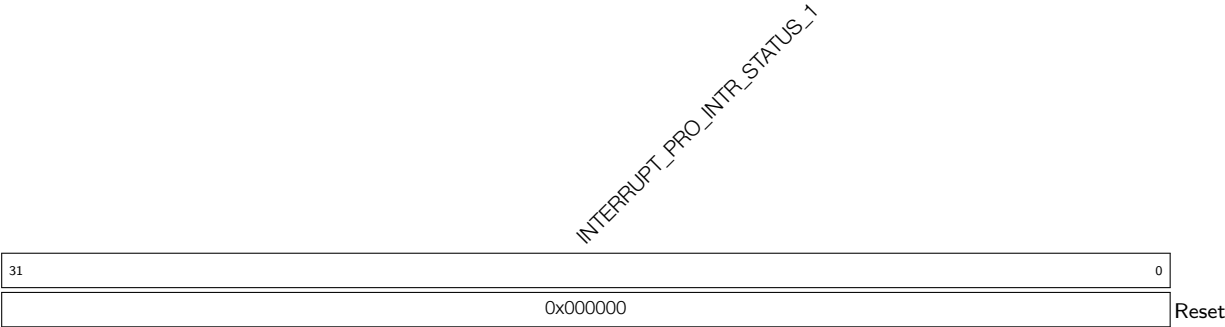
Register 4.97: INTERRUPT\_PRO\_INTR\_STATUS\_REG\_0\_REG (0x017C)



**INTERRUPT\_PRO\_INTR\_STATUS\_0** This register stores the status of the first 32 input interrupt sources. (RO)

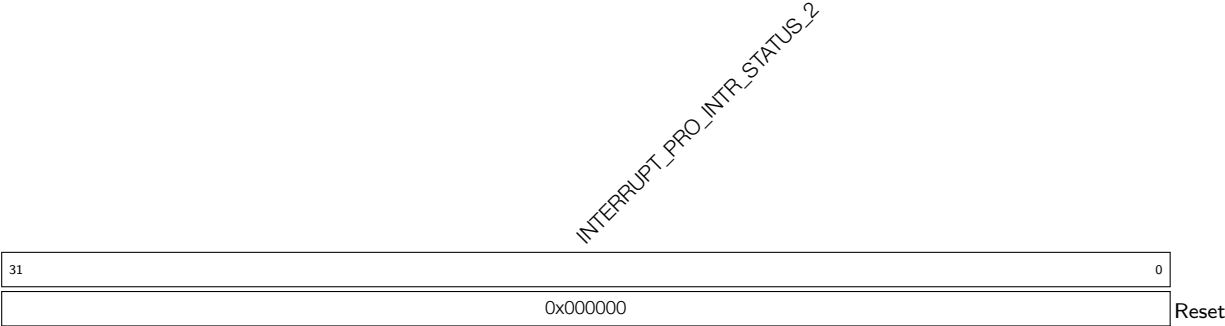


Register 4.98: INTERRUPT\_PRO\_INTR\_STATUS\_REG\_1\_REG (0x0180)



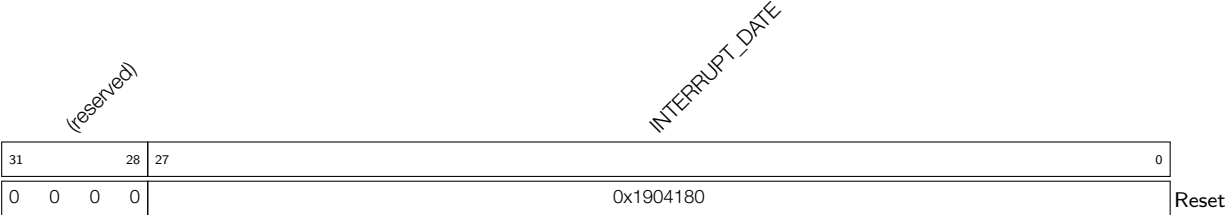
**INTERRUPT\_PRO\_INTR\_STATUS\_1** This register stores the status of the second 32 input interrupt sources. (RO)

Register 4.99: INTERRUPT\_PRO\_INTR\_STATUS\_REG\_2\_REG (0x0184)



**INTERRUPT\_PRO\_INTR\_STATUS\_2** This register stores the status of the last 31 input interrupt sources. (RO)

Register 4.100: INTERRUPT\_DATE\_REG (0x0FFC)



**INTERRUPT\_DATE** Version control register. (R/W)

## 5. IO MUX and GPIO Matrix

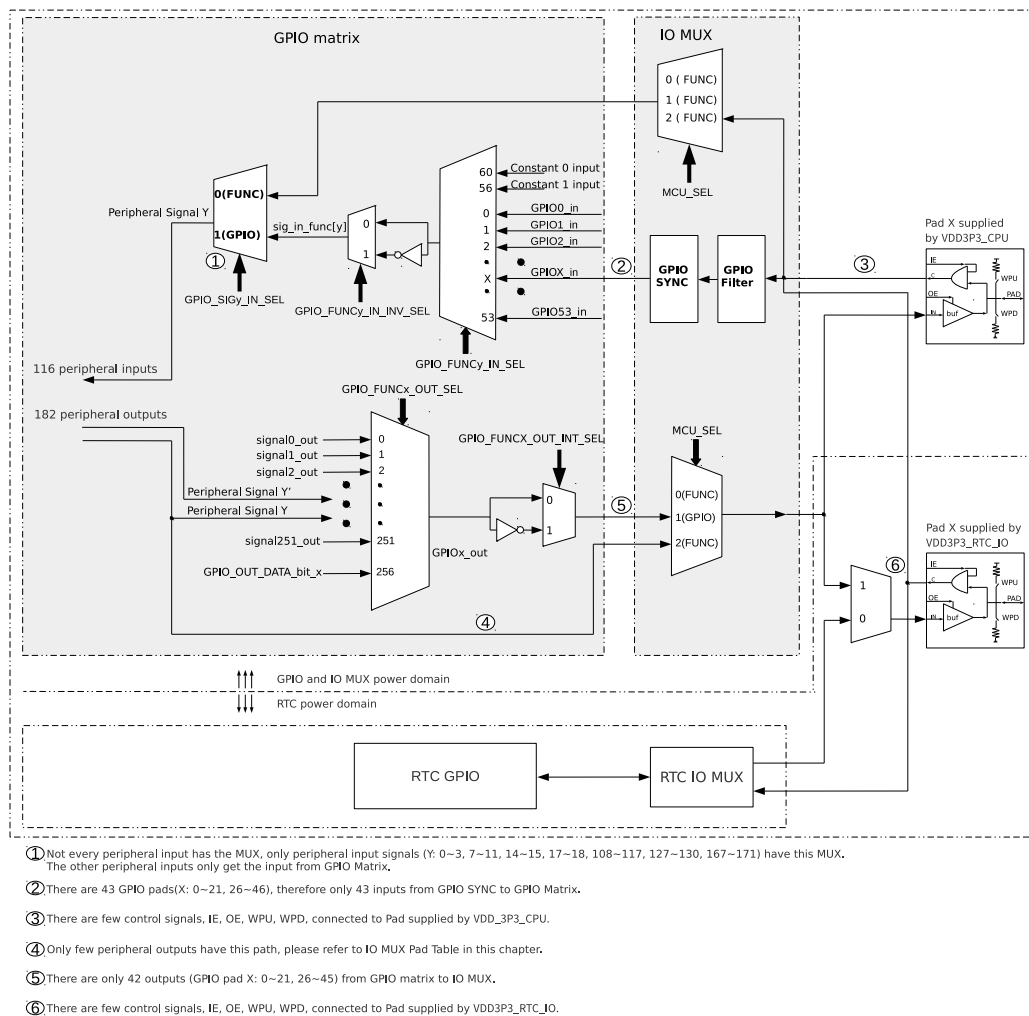
### 5.1 Overview

The ESP32-S2 chip features 43 physical GPIO pads. Each pad can be used as a general-purpose I/O, or be connected to an internal peripheral signal. The IO MUX, RTC IO MUX and the GPIO matrix are responsible for routing signals from the peripherals to GPIO pads. Together these modules provide highly configurable I/O.

**Note that the GPIO pads are numbered from 0 ~ 21 and 26 ~ 46, while GPIO46 is input-only.**

This chapter describes the selection and connection of the internal signals for the 43 digital pads and control signals: FUN\_SEL, IE, OE, WPU, WPD, etc. These internal signals include:

- 116 digital peripheral input signals, control signals: SIG\_IN\_SEL, SIG\_OUT\_SEL, IE, OE, etc.
- 182 digital peripheral output signals, control signals: SIG\_IN\_SEL, SIG\_OUT\_SEL, IE, OE, etc.
- fast peripheral input and output signals, control signals: IE, OE, etc.
- 22 RTC GPIO signals



**Figure 5-1. IO MUX, RTC IO MUX and GPIO Matrix Overview**

Figure 5-1 shows the overview of IO MUX, RTC IO MUX and GPIO matrix.

1. IO MUX provides one configuration register `IO_MUX_n_REG` for each GPIO pad. The pad can be configured to

- perform GPIO function routed by GPIO matrix;
- or perform direct connection bypassing GPIO matrix.

Some high-speed digital signals (SPI, JTAG, UART) can bypass GPIO matrix for better high-frequency digital performance. In this case, IO MUX is used to connect these pads directly to the peripheral.

See Section 5.11 for the IO MUX functions for each I/O pad.

2. GPIO matrix is a full-switching matrix between the peripheral input/output signals and the pads.
  - For input to the chip: each of the 116 internal peripheral inputs can select any GPIO pad as their input source.
  - For output from the chip: each GPIO pad can select any of the 182 peripheral output signals for its output.

See Section 5.10 for the list of peripheral signals via GPIO matrix.

3. RTC IO MUX is used to connect GPIO pads to their low-power and analog functions. Only a subset of GPIO pads have these optional RTC functions.

See Section 5.12 for the list of RTC IO MUX functions.

## 5.2 Peripheral Input via GPIO Matrix

### 5.2.1 Overview

To receive a peripheral input signal via GPIO matrix, the matrix is configured to source the peripheral input signal from one of the 43 GPIOs (0 ~ 21, 26 ~ 46), see Table 22. Meanwhile, register corresponding to the peripheral should be set to receive input signal via GPIO matrix.

### 5.2.2 Synchronization

When signals are directed using the GPIO matrix, the signal will be synchronized to the APB bus clock by the GPIO SYNC hardware. This synchronization applies to all GPIO matrix signals but does not apply when using the IO MUX, see Figure 5-1.

Figure 5-2 shows the functionality of GPIO SYNC. In the figure, negative sync and positive sync mean GPIO input is synchronized on APB clock falling edge and on APB clock rising edge, respectively.

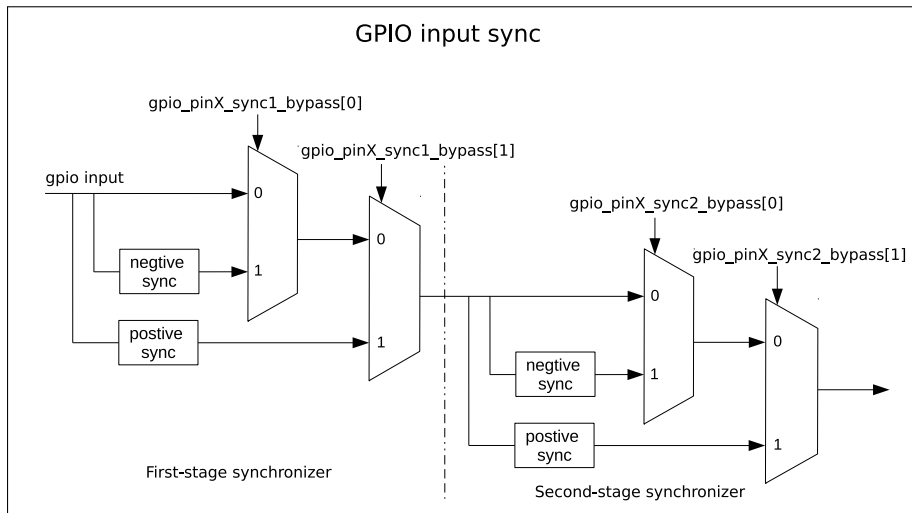


Figure 5-2. GPIO Input Synchronized on Clock Rising Edge or on Falling Edge

### 5.2.3 Functional Description

To read GPIO pad  $X$  into peripheral signal  $Y$ , follow the steps below:

1. Configure register `GPIO_FUNC $y$ _IN_SEL_CFG_REG` corresponding to peripheral signal  $Y$  in GPIO matrix:
  - Set `GPIO_SIG $y$ _IN_SEL` to enable peripheral signal input via GPIO matrix.
  - Set `GPIO_FUNC $y$ _IN_SEL` to the value corresponding to GPIO pad  $X$ .

**Note that** some peripheral signals have no valid `GPIO_SIG $y$ _IN_SEL` bit, namely, there is no MUX module in Figure 5-1 for these signals (see note 1 below Figure 5-1). These peripherals can only receive input signals via GPIO matrix.

2. Enable the filter for pad input signals by setting the register `IO_MUX_FILTER_EN`. Only the signals with a valid width of more than two clock cycles can be sampled, see Figure 5-3.

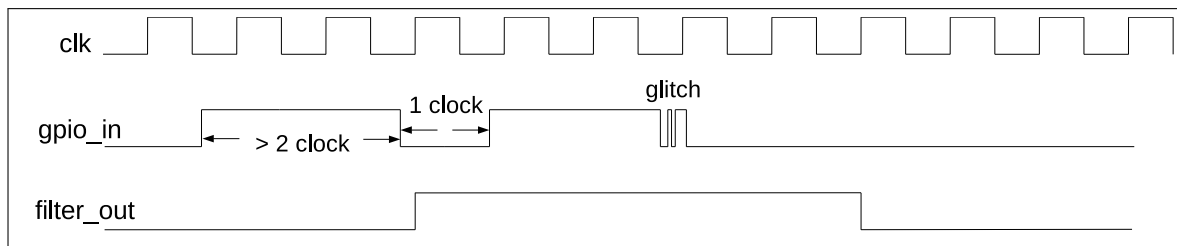


Figure 5-3. Filter Timing Diagram of GPIO Input Signals

3. Synchronize GPIO input. To do so, please set `GPIO_PIN $x$ _REG` corresponding to GPIO pad  $X$  as follows:
  - Set `GPIO_PIN $x$ _SYNC1_BYPASS` to enable input signal synchronized on rising edge or on falling edge in the first clock, see Figure 5-2.
  - Set `GPIO_PIN $x$ _SYNC2_BYPASS` to enable input signal synchronized on rising edge or on falling edge in the second clock, see Figure 5-2.
4. Configure IO MUX register to enable pad input. For this end, please set `IO_MUX_ $x$ _REG` corresponding to GPIO pad  $X$  as follows:
  - Set `IO_MUX_FUN_IE` to enable input.

- Set or clear `IO_MUX_FUN_WPU` and `IO_MUX_FUN_WPD`, as desired, to enable or disable pull-up and pull-down resistors.

For example, to connect RMT channel 0 input signal (`rmt_sig_in0`, signal index 83) to GPIO40, please follow the steps below. Note that GPIO40 is also named as MTDO pin.

1. Set `GPIO_SIG83_IN_SEL` in register `GPIO_FUNC83_IN_SEL_CFG_REG` to enable peripheral signal input via GPIO matrix.
2. Set `GPIO_FUNC83_IN_SEL` in register `GPIO_FUNC83_IN_SEL_CFG_REG` to 40.
3. Set `IO_MUX_FUN_IE` in register `IO_MUX_GPIO40_REG` to enable pad input.

**Note:**

- One input pad can be connected to multiple peripheral input signals.
- The input signal can be inverted by configuring `GPIO_FUNCy_IN_INV_SEL`.
- It is possible to have a peripheral read a constantly low or constantly high input value without connecting this input to a pad. This can be done by selecting a special `GPIO_FUNCy_IN_SEL` input, instead of a GPIO number:
  - When `GPIO_FUNCy_IN_SEL` is 0x3C, input signal `X` is always 0.
  - When `GPIO_FUNCy_IN_SEL` is 0x38, input signal `X` is always 1.

### 5.2.4 Simple GPIO Input

`GPIO_IN_REG/GPIO_IN1_REG` holds the input values of each GPIO pad. The input value of any GPIO pad can be read at any time without configuring GPIO matrix for a particular peripheral signal. However, it is necessary to enable the input in IO MUX by setting `IO_MUX_FUN_IE` bit in register `IO_MUX_n_REG` corresponding to pad `X`, as mentioned in Section 5.2.2.

## 5.3 Peripheral Output via GPIO Matrix

### 5.3.1 Overview

To output a signal from a peripheral via GPIO matrix, the matrix is configured to route peripheral output signals (0 ~ 11, 14 ~ 18, and etc.) to one of the 42 GPIOs (0 ~ 21, 26 ~ 45). See Table 22.

The output signal is routed from the peripheral into GPIO matrix and then into IO MUX. IO MUX must be configured to set the chosen pad to GPIO function. This causes the output GPIO signal to be connected to the pad.

**Note:**

There is a range of peripheral output signals (223 ~ 227) which are not connected to any peripheral. These can be used to input a signal from one GPIO pad and output directly to another GPIO pad.

### 5.3.2 Functional Description

Some of the 182 output signals can be set to go through GPIO matrix into IO MUX and then to a pad. Figure 5-1 illustrates the configuration.

To output peripheral signal `Y` to a particular GPIO pad `X`, follow these steps:

1. Configure register `GPIO_FUNCx_OUT_SEL_CFG_REG` and `GPIO_ENABLE_REG[x]` corresponding to GPIO pad `X` in GPIO matrix. Recommended operation: use corresponding `W1TS` (write 1 to set) and `W1TC` (write 1 to clear) registers to set or clear `GPIO_ENABLE_REG`.
  - Set the `GPIO_FUNCx_OUT_SEL` field in register `GPIO_FUNCx_OUT_SEL_CFG_REG` to the index of the desired peripheral output signal `Y`.
  - If the signal should always be enabled as an output, set the `GPIO_FUNCx_OEN_SEL` bit in register `GPIO_FUNCx_OUT_SEL_CFG_REG` and the bit in register `GPIO_ENABLE_W1TS_REG` or in register `GPIO_ENABLE1_W1TS_REG`, corresponding to GPIO pad `X`. To have the output enable signal decided by internal logic (see the column "Output enable of output signals" in Table 22), clear `GPIO_FUNCx_OEN_SEL` bit instead.
  - Clear the corresponding bit in register `GPIO_ENABLE_W1TC_REG` or in register `GPIO_ENABLE1_W1TC_REG` to disable the output from the GPIO pad.
2. For an open drain output, set the `GPIO_PINx_PAD_DRIVER` bit in register `GPIO_PINx_REG` corresponding to GPIO pad `X`.
3. Configure IO MUX register to enable output via GPIO matrix. Set the `IO_MUX_x_REG` corresponding to GPIO pad `X` as follows:
  - Set the field `IO_MUX_MCU_SEL` to IO\_MUX function corresponding to GPIO pad `X`. This is Function 1, numeric value 1, for all pins.
  - Set the `IO_MUX_FUN_DRV` field to the desired value for output strength (0 ~ 3). The higher the driver strength, the more current can be sourced/sunk from the pin.
    - 0: ~5 mA
    - 1: ~10 mA
    - 2: ~20 mA (Default value)
    - 3: ~40 mA
  - If using open drain mode, set/clear the `IO_MUX_FUN_WPU` and `IO_MUX_FUN_WPD` bits to enable/disable the internal pull-up/down resistors.

**Note:**

- The output signal from a single peripheral can be sent to multiple pads simultaneously.
- GPIO46 can not be used as an output.
- The output signal can be inverted by setting `GPIO_FUNCn_OUT_INV_SEL` bit.

### 5.3.3 Simple GPIO Output

GPIO matrix can also be used for simple GPIO output. This can be done as below:

- Set GPIO matrix `GPIO_FUNCn_OUT_SEL` with a special peripheral index 256 (0x100);
- Set the corresponding bit in `GPIO_OUT_REG[31:0]` or `GPIO_OUT1_REG[21:0]` register to the desired GPIO output value.

**Note:**

- `GPIO_OUT_REG[0] ~ GPIO_OUT_REG[31]` correspond to GPIO0 ~ GPIO31, and `GPIO_OUT1_REG[25:22]` are invalid.
- `GPIO_OUT1_REG[0] ~ GPIO_OUT1_REG[13]` correspond to GPIO32 ~ GPIO45, and `GPIO_OUT1_REG[21:14]` are invalid.
- Recommended operation: use corresponding W1TS and W1TC registers, such as `GPIO_OUT_W1TS/GPIO_OUT_W1TC` to set or clear the registers `GPIO_OUT_REG/GPIO_OUT1_REG`.

### 5.3.4 Sigma Delta Modulated Output

#### 5.3.4.1 Functional Description

ESP32-S2 provides a second-order sigma delta modulation module and eight independent modulation channels. The channels are capable to output 1-bit signals (output index: 100 ~ 107) with sigma delta modulation, and by default output is enabled for these channels. This module can also output PDM (pulse density modulation) signal with configurable duty cycle. The transfer function is:

$$H(z) = X(z)z^{-1} + E(z)(1-z^{-1})^2$$

$E(z)$  is quantization error and  $X(z)$  is the input.

Sigma Delta modulator supports scaling down of APB\_CLK by divider 1 ~ 256:

- Set `GPIOSD_FUNCTION_CLK_EN` to enable the modulator clock.
- Configure register `GPIOSD_SDn_PRESCALE` ( $n$  is 0 ~ 7 for eight channels).

After scaling, the clock cycle is equal to one pulse output cycle from the modulator.

`GPIOSD_SDn_IN` is a signed number with a range of [-128, 127] and is used to control the duty cycle<sup>1</sup> of PDM output signal.

- `GPIOSD_SDn_IN` = -128, the duty cycle of the output signal is 0%.
- `GPIOSD_SDn_IN` = 0, the duty cycle of the output signal is near 50%.
- `GPIOSD_SDn_IN` = 127, the duty cycle of the output signal is close to 100%.

The formula for calculating PDM signal duty cycle is shown as below:

$$Duty\_Cycle = \frac{GPIOSD\_SDn\_IN + 128}{256}$$

**Note:**

For PDM signals, duty cycle refers to the percentage of high level cycles to the whole statistical period (several pulse cycles, for example 256 pulse cycles).

#### 5.3.4.2 SDM Configuration

The configuration of SDM is shown below:

- Route one of SDM outputs to a pad via GPIO matrix, see Section 5.3.2.
- Enable the modulator clock by setting the register `GPIOSD_FUNCTION_CLK_EN`.

- Configure the divider value by setting the register `GPIOSD_SDn_PRESCALE`.
- Configure the duty cycle of SDM output signal by setting the register `GPIOSD_SDn_IN`.

## 5.4 Dedicated GPIO

### 5.4.1 Overview

The dedicated GPIO module, consisting of eight input/output channels, is specially designed for CPU interaction with GPIO matrix and IO MUX. Peripheral input/output signals for input/output channels both are indexed from 235 to 242. By default, the output is enabled for output channels.

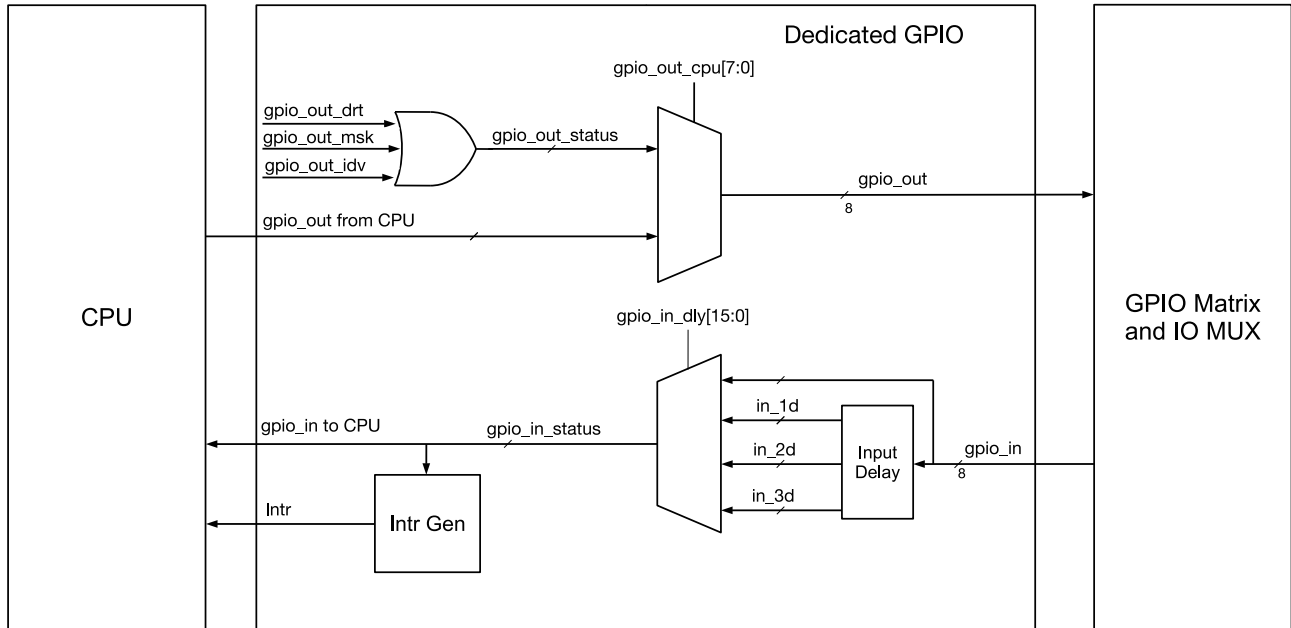


Figure 5-4. Dedicated GPIO Diagram

Figure 5-4 shows the structure of dedicated GPIO module. Users can enable the clock of this module by setting the bit `SYSTEM_CLK_EN_DEDICATED_GPIO` in register `SYSTEM_CPU_PERI_CLK_EN_REG`, and reset this module by setting the bit `SYSTEM_RST_EN_DEDICATED_GPIO` in register `SYSTEM_CPU_PERI_RST_EN_REG` first, and then clearing this bit. For more information, please refer to Table 33 *Peripheral Clock Gating and Reset Bits* in Chapter 6 *System Registers*.

### 5.4.2 Features

Dedicated GPIO module has the following features:

- Eight output and eight input channels
- Each channel accessible with registers or directly by CPU instructions
- Configurable delay on input channels
- Interrupts on input channels



### 5.4.3 Functional Description

Dedicated GPIOs may be accessed using registers or directly by calling specific CPU instructions.

When accessing output channels, select between registers and CPU by configuring `DEDIC_GPIO_OUT_CPU_REG`:

- `DEDIC_GPIO_OUT_CPU_SELn` = 0, drive GPIO output via registers.
- `DEDIC_GPIO_OUT_CPU_SELn` = 1, drive GPIO output via CPU instructions.

The dedicated GPIO module also provides two ways to read input channels:

- Query GPIO input value via registers.
- Read GPIO input value via CPU instructions.

#### 5.4.3.1 Accessing GPIO via Registers

Users can control GPIO output via registers in the following ways:

- Write GPIO output value directly by configuring the register `DEDIC_GPIO_OUT_DRT_REG`.
- Write GPIO output value via masked access by configuring the register `DEDIC_GPIO_OUT_MSK_REG`.
- Write GPIO output value via individual bits by configuring the register `DEDIC_GPIO_OUT_IDV_REG`.

User can read the register `DEDIC_GPIO_OUT_SCAN_REG` to check GPIO status, i.e. `gpio_out_status` in Figure 5-4, via software.

Users can get a dedicated GPIO input value by reading the register `DEDIC_GPIO_IN_SCAN_REG`, i.e. the `gpio_in_status` in Figure 5-4, via software.

The dedicated GPIO module supports for a delay of 1/2/3 clock cycle(s) for input signals, or with no delay, which can be controlled by configuring the register `DEDIC_GPIO_IN_DLY_REG` for each individual channel. GPIO input status is indicated by interrupts. Users can configure the register `DEDIC_GPIO_INTR_RCGN_REG` to set trigger modes:

- 0/1: no interrupt
- 2: low level trigger
- 3: high level trigger
- 4: falling edge trigger
- 5: rising edge trigger
- 6/7: edges trigger

#### 5.4.3.2 Accessing GPIO with CPU

CPU can also read/write a dedicated GPIO via instructions.

- Set bits in output channel.  
Assembly syntax: `SET_BIT_GPIO_OUT mask`  
Addressing: immediate addressing

Function: write 1 to set the corresponding bits in user register GPIO\_OUT, i.e. the gpio\_out in Figure 5-4. The “mask” is 8 bits wide. The bits in GPIO\_OUT, corresponding to the bits in “mask” with the value of 1, will be set to 1, while the other bits in GPIO\_OUT remain unaffected.

- Clear bits in output channel.

Assembly syntax: CLR\_BIT\_GPIO\_OUT mask

Addressing: immediate addressing

Function: write 1 to clear the corresponding bits in user register GPIO\_OUT. The “mask” is 8 bits wide. The bits in GPIO\_OUT, corresponding to the bits in “mask” with the value of 1, will be cleared, while the other bits in GPIO\_OUT remain unaffected.

- Set or clear bits in output channel with masked access

Assembly syntax: WR\_MASK\_GPIO\_OUT value, mask

Addressing: register addressing

Function: write value to user register GPIO\_OUT via masked access. The “value” is 8 bits wide, which represents the value to write. The “mask” is 8 bits wide, which represents the bits in GPIO\_OUT to be manipulated. For example, mask 0x03 (0000 0011) indicates that the write value is only valid for GPIO\_OUT[0] and GPIO\_OUT[1]. Only the bits in GPIO\_OUT, corresponding to the bits with the value of 1 in the “mask”, are updated, that is, updated to the value stored in “value”.

- Write “art” register to the output channel.

Assembly syntax: WUR.GPIO\_OUT art

Addressing: register addressing

Function: write the value of the address register “art” to the user register GPIO\_OUT. Register “art” is 32 bits wide, and only low 8 bits are valid when using this instruction.

- Read the output channel to “arr” register.

Assembly syntax: RUR.GPIO\_OUT arr

Addressing: register addressing

Function: read the value of the user register GPIO\_OUT to the address register “arr”. Register “arr” is 32 bits wide, and only low 8 bits are valid when using this instruction.

- Read the input channel to “l” register.

Assembly syntax: GET\_GPIO\_IN l

Addressing: register addressing

Function: read the value of the user register GPIO\_IN, i.e. the gpio\_in in Figure 5-4, to the address register “l”. Register “l” is 32 bits wide, the high 24 bits of which are 0, and low 8 bits of which corresponds to the value of the user register GPIO\_IN.

## 5.5 Direct I/O via IO MUX

### 5.5.1 Overview

Some high-speed signals (SPI and JTAG) can bypass GPIO matrix for better high-frequency digital performance. In this case, IO MUX is used to connect these pads directly to the peripheral.

This option is less flexible than routing signals via GPIO matrix, as the IO MUX register for each GPIO pad can only select from a limited number of functions, but high-frequency digital performance can be improved.

### 5.5.2 Functional Description

Two registers must be configured in order to bypass GPIO matrix for peripheral input signals:

1. [IO\\_MUX\\_MCU\\_SEL](#) for the GPIO pad must be set to the required pad function. For the list of pad functions, please refer to Section 5.11.
2. Set [GPIO\\_SIG<sub>n</sub>\\_IN\\_SEL](#) to low level to route the input directly to the peripheral.

To bypass GPIO matrix for peripheral output signals, [IO\\_MUX\\_MCU\\_SEL](#) for the GPIO pad must be set to the required pad function. For the list of pad functions, please refer to Section 5.11.

#### Note:

For peripheral I/O signals, not all signals can be connected to peripheral via IO MUX. Some specific input signals and some specific output signals can only be connected to peripheral via GPIO matrix.

## 5.6 RTC IO MUX for Low Power and Analog I/O

### 5.6.1 Overview

22 GPIO pads have low power capabilities (RTC domain) and analog functions which are handled by the RTC subsystem of ESP32-S2. IO MUX and GPIO matrix are not used for these functions, rather, RTC IO MUX is used to redirect input/output signals to the RTC subsystem.

When configured as RTC GPIOs, the output pads can still retain the output level value when the chip is in Deep-sleep mode, and the input pads can wake up the chip from Deep-sleep.

Section 5.12 lists the RTC\_MUX pins and their functions.

### 5.6.2 Functional Description

Each pad with analog and RTC functions is controlled by [RTCIO\\_TOUCH\\_PAD<sub>n</sub>\\_MUX\\_SEL](#) bit in register [RTCIO\\_TOUCH\\_PAD<sub>n</sub>\\_REG](#). By default all bits in these registers are set to 0, routing all input/output signals via IO MUX.

If [RTCIO\\_TOUCH\\_PAD<sub>n</sub>\\_MUX\\_SEL](#) is set to 1, then input/output signals to and from that pad is routed to the RTC subsystem. In this mode, [RTCIO\\_TOUCH\\_PAD<sub>n</sub>\\_REG](#) is used for digital input/output and the analog features of the pad are also available.

Please refer to Section 5.12 for the list of RTC pin functions and the mapping table of GPIO pads to their analog functions. Note that [RTCIO\\_TOUCH\\_PAD<sub>n</sub>\\_REG](#) applies the RTC GPIO pin numbering, not the GPIO pad numbering.

## 5.7 Pin Functions in Light-sleep

Pins may provide different functions when ESP32-S2 is in Light-sleep mode. If [IO\\_MUX\\_SLP\\_SEL](#) in register [IO\\_MUX<sub>n</sub>\\_REG](#) for a GPIO pad is set to 1, a different set of bits will be used to control the pad when the chip is in Light-sleep mode.

**Table 21: Pin Function Register for IO MUX Light-sleep Mode**

IO MUX Function	Normal Execution OR <a href="#">IO_MUX_SLP_SEL</a> = 0	Light-sleep Mode AND <a href="#">IO_MUX_SLP_SEL</a> = 1
Output Drive Strength	<a href="#">IO_MUX_FUN_DRV</a>	<a href="#">IO_MUX_FUN_DRV</a>

Pullup Resistor	<a href="#">IO_MUX_FUN_WPU</a>	<a href="#">IO_MUX_MCU_WPU</a>
Pulldown Resistor	<a href="#">IO_MUX_FUN_WPD</a>	<a href="#">IO_MUX_MCU_WPD</a>
Output Enable	(From GPIO Matrix <code>_OEN</code> field) <sup>1</sup>	<a href="#">IO_MUX_MCU_OE</a>

If [IO\\_MUX\\_SLP\\_SEL](#) is set to 0, pin functions remain the same in both normal execution and Light-sleep mode.

**Note:**

Please refer to Section [5.3.2](#) for how to enable output in normal execution (when [IO\\_MUX\\_SLP\\_SEL](#) = 0).

## 5.8 Pad Hold Feature

Each IO pad (including the RTC pads) has an individual hold function controlled by a RTC register. When the pad is set to hold, the state is latched at that moment and will not change no matter how the internal signals change or how the IO MUX/GPIO configuration is modified. Users can use the hold function for the pads to retain the pad state through a core reset and system reset triggered by watchdog time-out or Deep-sleep events.

**Note:**

- For digital pads, to maintain pad input/output status in Deep-sleep mode, users can set [RTC\\_CNTL\\_DG\\_PAD\\_FORCE\\_UNHOLD](#) to 0 before powering down. For RTC pads, the input and output values are controlled by the corresponding bits of register [RTC\\_CNTL\\_PAD\\_HOLD\\_REG](#), and users can set it to 1 to hold the value or set it to 0 to unhold the value.
- For digital pads, to disable the hold function after the chip is woken up, users can set [RTC\\_CNTL\\_DG\\_PAD\\_FORCE\\_UNHOLD](#) to 1. To maintain the hold function of the pad, users can change the corresponding bit in register [RTC\\_CNTL\\_PAD\\_HOLD\\_REG](#) to 1.

## 5.9 I/O Pad Power Supplies

For more information on the power supply for IO pads, please refer to Pin Definition in [ESP32-S2 Datasheet](#).

### 5.9.1 Power Supply Management

Each ESP32-S2 digital pin is connected to one of the four different power domains.

- VDD3P3\_RTC\_IO: the input power supply for both RTC and CPU
- VDD3P3\_CPU: the input power supply for CPU
- VDD3P3\_RTC: the input power supply for RTC analog part
- VDD\_SPI: configurable power supply

VDD\_SPI can be configured to use an internal LDO. The LDO input is VDD3P3\_RTC\_IO and the output is 1.8 V. If the LDO is not enabled, VDD\_SPI is connected directly to the same power supply as VDD3P3\_RTC\_IO.

The VDD\_SPI configuration is determined by the value of strapping pin GPIO45, or can be overridden by eFuse and/or register settings. See [ESP32-S2 Datasheet](#) sections Power Scheme and Strapping Pins for more details.

## 5.10 Peripheral Signal List

Table 22 shows the peripheral input/output signals via GPIO matrix.

**Table 22: GPIO Matrix**

Signal No.	Input signals	Default value if unassigned *	Same input signal from IO MUX core	Output signals	Output enable of output signals
0	SPIQ_in	0	yes	SPIQ_out	SPIQ_oe
1	SPID_in	0	yes	SPID_out	SPID_oe
2	SPIHD_in	0	yes	SPIHD_out	SPIHD_oe
3	SPIWP_in	0	yes	SPIWP_out	SPIWP_oe
4	-	-	-	SPICLK_out_mux	SPICLK_oe
5	-	-	-	SPICS0_out	SPICS0_oe
6	-	-	-	SPICS1_out	SPICS1_oe
7	SPID4_in	0	yes	SPID4_out	SPID4_oe
8	SPID5_in	0	yes	SPID5_out	SPID5_oe
9	SPID6_in	0	yes	SPID6_out	SPID6_oe
10	SPID7_in	0	yes	SPID7_out	SPID7_oe
11	SPIDQS_in	0	yes	SPIDQS_out	SPIDQS_oe
14	U0RXD_in	0	yes	U0TXD_out	1'd1
15	U0CTS_in	0	yes	U0RTS_out	1'd1
16	U0DSR_in	0	no	U0DTR_out	1'd1
17	U1RXD_in	0	yes	U1TXD_out	1'd1
18	U1CTS_in	0	yes	U1RTS_out	1'd1
21	U1DSR_in	0	no	U1DTR_out	1'd1
23	I2S0O_BCK_in	0	no	I2S0O_BCK_out	1'd1
25	I2S0O_WS_in	0	no	I2S0O_WS_out	1'd1
27	I2S0I_BCK_in	0	no	I2S0I_BCK_out	1'd1
28	I2S0I_WS_in	0	no	I2S0I_WS_out	1'd1
29	I2CEXT0_SCL_in	1	no	I2CEXT0_SCL_out	I2CEXT0_SCL_oe
30	I2CEXT0_SDA_in	1	no	I2CEXT0_SDA_out	I2CEXT0_SDA_oe
39	pcnt_sig_ch0_in0	0	no	gpio_wlan_prio	1'd1
40	pcnt_sig_ch1_in0	0	no	gpio_wlan_active	1'd1
41	pcnt_ctrl_ch0_in0	0	no	-	1'd1
42	pcnt_ctrl_ch1_in0	0	no	-	1'd1
43	pcnt_sig_ch0_in1	0	no	-	1'd1
44	pcnt_sig_ch1_in1	0	no	-	1'd1
45	pcnt_ctrl_ch0_in1	0	no	-	1'd1
46	pcnt_ctrl_ch1_in1	0	no	-	1'd1
47	pcnt_sig_ch0_in2	0	no	-	1'd1
48	pcnt_sig_ch1_in2	0	no	-	1'd1
49	pcnt_ctrl_ch0_in2	0	no	-	1'd1
50	pcnt_ctrl_ch1_in2	0	no	-	1'd1
51	pcnt_sig_ch0_in3	0	no	-	1'd1

Signal No.	Input signals	Default value if unassigned *	Same input signal from IO MUX core	Output signals	Output enable of output signals
52	pcnt_sig_ch1_in3	0	no	-	1'd1
53	pcnt_ctrl_ch0_in3	0	no	-	1'd1
54	pcnt_ctrl_ch1_in3	0	no	-	1'd1
64	usb_otg_iddig_in	0	no	-	1'd1
65	usb_otg_avalid_in	0	no	-	1'd1
66	usb_srp_bvalid_in	0	no	usb_otg_idpullup	1'd1
67	usb_otg_vbusvalid_in	0	no	usb_otg_dppulldown	1'd1
68	usb_srp_sessend_in	0	no	usb_otg_dmpulldown	1'd1
69	-	-	-	usb_otg_drvvbus	1'd1
70	-	-	-	usb_srp_chrgvbus	1'd1
71	-	-	-	usb_srp_dischrgvbus	1'd1
72	SPI3_CLK_in	0	no	SPI3_CLK_out_mux	SPI3_CLK_oe
73	SPI3_Q_in	0	no	SPI3_Q_out	SPI3_Q_oe
74	SPI3_D_in	0	no	SPI3_D_out	SPI3_D_oe
75	SPI3_HD_in	0	no	SPI3_HD_out	SPI3_HD_oe
76	SPI3_CS0_in	0	no	SPI3_CS0_out	SPI3_CS0_oe
77	-	-	-	SPI3_CS1_out	SPI3_CS1_oe
78	-	-	-	SPI3_CS2_out	SPI3_CS2_oe
79	-	-	-	ledc_ls_sig_out0	1'd1
80	-	-	-	ledc_ls_sig_out1	1'd1
81	-	-	-	ledc_ls_sig_out2	1'd1
82	-	-	-	ledc_ls_sig_out3	1'd1
83	rmt_sig_in0	0	no	ledc_ls_sig_out4	1'd1
84	rmt_sig_in1	0	no	ledc_ls_sig_out5	1'd1
85	rmt_sig_in2	0	no	ledc_ls_sig_out6	1'd1
86	rmt_sig_in3	0	no	ledc_ls_sig_out7	1'd1
87	-	-	-	rmt_sig_out0	1'd1
88	-	-	-	rmt_sig_out1	1'd1
89	-	-	-	rmt_sig_out2	1'd1
90	-	-	-	rmt_sig_out3	1'd1
95	I2CEXT1_SCL_in	1	no	I2CEXT1_SCL_out	I2CEXT1_SCL_oe
96	I2CEXT1_SDA_in	1	no	I2CEXT1_SDA_out	I2CEXT1_SDA_oe
100	-	-	-	gpio_sd0_out	1'd1
101	-	-	-	gpio_sd1_out	1'd1
102	-	-	-	gpio_sd2_out	1'd1
103	-	-	-	gpio_sd3_out	1'd1
104	-	-	-	gpio_sd4_out	1'd1
105	-	-	-	gpio_sd5_out	1'd1
106	-	-	-	gpio_sd6_out	1'd1
107	-	-	-	gpio_sd7_out	1'd1
108	FSPICLK_in	0	yes	FSPICLK_out_mux	FSPICLK_oe

Signal No.	Input signals	Default value if unassigned *	Same input signal from IO MUX core	Output signals	Output enable of output signals
109	FSPIQ_in	0	yes	FSPIQ_out	FSPIQ_oe
110	FSPID_in	0	yes	FSPID_out	FSPID_oe
111	FSPiHD_in	0	yes	FSPiHD_out	FSPiHD_oe
112	FSPiWP_in	0	yes	FSPiWP_out	FSPiWP_oe
113	FSPiIO4_in	0	yes	FSPiIO4_out	FSPiIO4_oe
114	FSPiIO5_in	0	yes	FSPiIO5_out	FSPiIO5_oe
115	FSPiIO6_in	0	yes	FSPiIO6_out	FSPiIO6_oe
116	FSPiIO7_in	0	yes	FSPiIO7_out	FSPiIO7_oe
117	FSPiCS0_in	0	yes	FSPiCS0_out	FSPiCS0_oe
118	-	-	-	FSPiCS1_out	FSPiCS1_oe
119	-	-	-	FSPiCS2_out	FSPiCS2_oe
120	-	-	-	FSPiCS3_out	FSPiCS3_oe
121	-	-	-	FSPiCS4_out	FSPiCS4_oe
122	-	-	-	FSPiCS5_out	FSPiCS5_oe
123	can_rx	1	no	can_tx	1'd1
124	-	-	-	can_bus_off_on	1'd1
125	-	-	-	can_clkout	1'd1
126	-	-	-	SUBSPiCLK_out_mux	SUBSPiCLK_oe
127	SUBSPiQ_in	0	yes	SUBSPiQ_out	SUBSPiQ_oe
128	SUBSPiD_in	0	yes	SUBSPiD_out	SUBSPiD_oe
129	SUBSPiHD_in	0	yes	SUBSPiHD_out	SUBSPiHD_oe
130	SUBSPiWP_in	0	yes	SUBSPiWP_out	SUBSPiWP_oe
131	-	-	-	SUBSPiCS0_out	SUBSPiCS0_oe
132	-	-	-	SUBSPiCS1_out	SUBSPiCS1_oe
133	-	-	-	FSPiDQS_out	FSPiDQS_oe
134	-	-	-	FSPi_HSYNC_out	FSPi_HSYNC_oe
135	-	-	-	FSPi_VSYNC_out	FSPi_VSYNC_oe
136	-	-	-	FSPi_DE_out	FSPi_DE_oe
137	-	-	-	FSPiCD_out	FSPiCD_oe
139	-	-	-	SPI3_CD_out	SPI3_CD_oe
140	-	-	-	SPI3_DQS_out	SPI3_DQS_oe
143	I2S0I_DATA_in0	0	no	I2S0O_DATA_out0	1'd1
144	I2S0I_DATA_in1	0	no	I2S0O_DATA_out1	1'd1
145	I2S0I_DATA_in2	0	no	I2S0O_DATA_out2	1'd1
146	I2S0I_DATA_in3	0	no	I2S0O_DATA_out3	1'd1
147	I2S0I_DATA_in4	0	no	I2S0O_DATA_out4	1'd1
148	I2S0I_DATA_in5	0	no	I2S0O_DATA_out5	1'd1
149	I2S0I_DATA_in6	0	no	I2S0O_DATA_out6	1'd1
150	I2S0I_DATA_in7	0	no	I2S0O_DATA_out7	1'd1
151	I2S0I_DATA_in8	0	no	I2S0O_DATA_out8	1'd1
152	I2S0I_DATA_in9	0	no	I2S0O_DATA_out9	1'd1

Signal No.	Input signals	Default value if unassigned *	Same input signal from IO MUX core	Output signals	Output enable of output signals
153	I2S0I_DATA_in10	0	no	I2S0O_DATA_out10	1'd1
154	I2S0I_DATA_in11	0	no	I2S0O_DATA_out11	1'd1
155	I2S0I_DATA_in12	0	no	I2S0O_DATA_out12	1'd1
156	I2S0I_DATA_in13	0	no	I2S0O_DATA_out13	1'd1
157	I2S0I_DATA_in14	0	no	I2S0O_DATA_out14	1'd1
158	I2S0I_DATA_in15	0	no	I2S0O_DATA_out15	1'd1
159	-	-	-	I2S0O_DATA_out16	1'd1
160	-	-	-	I2S0O_DATA_out17	1'd1
161	-	-	-	I2S0O_DATA_out18	1'd1
162	-	-	-	I2S0O_DATA_out19	1'd1
163	-	-	-	I2S0O_DATA_out20	1'd1
164	-	-	-	I2S0O_DATA_out21	1'd1
165	-	-	-	I2S0O_DATA_out22	1'd1
166	-	-	-	I2S0O_DATA_out23	1'd1
167	SUBSPID4_in	0	yes	SUBSPID4_out	SUBSPID4_oe
168	SUBSPID5_in	0	yes	SUBSPID5_out	SUBSPID5_oe
169	SUBSPID6_in	0	yes	SUBSPID6_out	SUBSPID6_oe
170	SUBSPID7_in	0	yes	SUBSPID7_out	SUBSPID7_oe
171	SUBSPIDQS_in	0	yes	SUBSPIDQS_out	SUBSPIDQS_oe
193	I2S0I_H_SYNC	0	no	-	1'd1
194	I2S0I_V_SYNC	0	no	-	1'd1
195	I2S0I_H_ENABLE	0	no	-	1'd1
215	-	-	-	ant_sel0	1'd1
216	-	-	-	ant_sel1	1'd1
217	-	-	-	ant_sel2	1'd1
218	-	-	-	ant_sel3	1'd1
219	-	-	-	ant_sel4	1'd1
220	-	-	-	ant_sel5	1'd1
221	-	-	-	ant_sel6	1'd1
222	-	-	-	ant_sel7	1'd1
223	sig_in_func_223	0	no	sig_in_func223	1'd1
224	sig_in_func_224	0	no	sig_in_func224	1'd1
225	sig_in_func_225	0	no	sig_in_func225	1'd1
226	sig_in_func_226	0	no	sig_in_func226	1'd1
227	sig_in_func_227	0	no	sig_in_func227	1'd1
235	pro_alonegpio_in0	0	no	pro_alonegpio_out0	1'd1
236	pro_alonegpio_in1	0	no	pro_alonegpio_out1	1'd1
237	pro_alonegpio_in2	0	no	pro_alonegpio_out2	1'd1
238	pro_alonegpio_in3	0	no	pro_alonegpio_out3	1'd1
239	pro_alonegpio_in4	0	no	pro_alonegpio_out4	1'd1
240	pro_alonegpio_in5	0	no	pro_alonegpio_out5	1'd1



Signal No.	Input signals	Default value if unassigned *	Same input signal from IO MUX core	Output signals	Output enable of output signals
241	pro_alonegpio_in6	0	no	pro_alonegpio_out6	1'd1
242	pro_alonegpio_in7	0	no	pro_alonegpio_out7	1'd1
251	-	-	-	clk_i2s_mux	1'd1

## 5.11 IO MUX Pad List

Table 23 shows the IO MUX functions of each I/O pad:

**Table 23: IO MUX Pad List**

GPIO	Pad Name	Function 0	Function 1	Function 2	Function 3	Function 4	Reset	Notes
0	GPIO0	GPIO0	GPIO0	-	-	-	3	R
1	GPIO1	GPIO1	GPIO1	-	-	-	1	R
2	GPIO2	GPIO2	GPIO2	-	-	-	1	R
3	GPIO3	GPIO3	GPIO3	-	-	-	0	R
4	GPIO4	GPIO4	GPIO4	-	-	-	0	R
5	GPIO5	GPIO5	GPIO5	-	-	-	0	R
6	GPIO6	GPIO6	GPIO6	-	-	-	0	R
7	GPIO7	GPIO7	GPIO7	-	-	-	0	R
8	GPIO8	GPIO8	GPIO8	-	SUBSPICS1	-	0	R
9	GPIO9	GPIO9	GPIO9	-	SUBSPIHD	FSPiHD	1	R
10	GPIO10	GPIO10	GPIO10	FSPIIO4	SUBSPICS0	FSPICS0	1	R
11	GPIO11	GPIO11	GPIO11	FSPIIO5	SUBSPID	FSPID	1	R
12	GPIO12	GPIO12	GPIO12	FSPIIO6	SUBSPICLK	FSPICLK	1	R
13	GPIO13	GPIO13	GPIO13	FSPIIO7	SUBSPIQ	FSPIQ	1	R
14	GPIO14	GPIO14	GPIO14	FSPIDQS	SUBSPIWP	FSPiWP	1	R
15	XTAL_32K_P	XTAL_32K_P	GPIO15	U0RTS	-	-	0	R
16	XTAL_32K_N	XTAL_32K_N	GPIO16	U0CTS	-	-	0	R
17	DAC_1	DAC_1	GPIO17	U1TXD	-	-	1	R
18	DAC_2	DAC_2	GPIO18	U1RXD	CLK_OUT3	-	1	R
19	GPIO19	GPIO19	GPIO19	U1RTS	CLK_OUT2	-	0	R
20	GPIO20	GPIO20	GPIO20	U1CTS	CLK_OUT1	-	0	R
21	GPIO21	GPIO21	GPIO21	-	-	-	0	R
26	SPICS1	SPICS1	GPIO26	-	-	-	3	-
27	SPIHD	SPIHD	GPIO27	-	-	-	3	-
28	SPIWP	SPIWP	GPIO28	-	-	-	3	-
29	SPICS0	SPICS0	GPIO29	-	-	-	3	-
30	SPICLK	SPICLK	GPIO30	-	-	-	3	-
31	SPIQ	SPIQ	GPIO31	-	-	-	3	-
32	SPID	SPID	GPIO32	-	-	-	3	-
33	GPIO33	GPIO33	GPIO33	FSPIHD	SUBSPIHD	SPIIO4	1	-
34	GPIO34	GPIO34	GPIO34	FSPICS0	SUBSPICS0	SPIIO5	1	-
35	GPIO35	GPIO35	GPIO35	FSPID	SUBSPID	SPIIO6	1	-

GPIO	Pad Name	Function 0	Function 1	Function 2	Function 3	Function 4	Reset	Notes
36	GPIO36	GPIO36	GPIO36	FSPICLK	SUBSPICLK	SPIIO7	1	-
37	GPIO37	GPIO37	GPIO37	FSPIQ	SUBSPIQ	SPIDQS	1	-
38	GPIO38	GPIO38	GPIO38	FSPIWP	SUBSPIWP	-	1	-
39	MTCK	MTCK	GPIO39	CLK_OUT3	SUBSPICS1	-	1	-
40	MTDO	MTDO	GPIO40	CLK_OUT2	-	-	1	-
41	MTDI	MTDI	GPIO41	CLK_OUT1	-	-	1	-
42	MTMS	MTMS	GPIO42	-	-	-	1	-
43	U0TXD	U0TXD	GPIO43	CLK_OUT1	-	-	3	-
44	U0RXD	U0RXD	GPIO44	CLK_OUT2	-	-	3	-
45	GPIO45	GPIO45	GPIO45	-	-	-	2	-
46	GPIO46	GPIO46	GPIO46	-	-	-	2	I

### Reset Configurations

“Reset” column shows the default configuration of each pad after reset:

- **0** - IE=0 (input disabled)
- **1** - IE=1 (input enabled)
- **2** - IE=1, WPD=1 (input enabled, pull-down resistor enabled)
- **3** - IE=1, WPU=1 (input enabled, pull-up resistor enabled)

#### Note:

- **R** - Pad has RTC/analog functions via RTC IO MUX.
- **I** - Pad can only be configured as input GPIO.

Please refer to Appendix A – ESP32-S2 Pin Lists in [ESP32-S2 Datasheet](#) for more details.

## 5.12 RTC IO MUX Pin List

Table 24 shows the RTC pins and how they correspond to GPIO pads.

**Table 24: RTC IO MUX Pin Summary**

RTC GPIO Num	GPIO Num	Pad Name	Analog Function			
			1	2	3	4
0	0	TOUCH_PAD0*	RTC_GPIO0	-	-	sar_i2c_scl_0
1	1	TOUCH_PAD1	RTC_GPIO1	-	-	sar_i2c_sda_0
2	2	TOUCH_PAD2	RTC_GPIO2	-	-	sar_i2c_scl_1
3	3	TOUCH_PAD3	RTC_GPIO3	-	-	sar_i2c_sda_1
4	4	TOUCH_PAD4	RTC_GPIO4	-	-	-
5	5	TOUCH_PAD5	RTC_GPIO5	-	-	-
6	6	TOUCH_PAD6	RTC_GPIO6	-	-	-
7	7	TOUCH_PAD7	RTC_GPIO7	-	-	-
8	8	TOUCH_PAD8	RTC_GPIO8	-	-	-
9	9	TOUCH_PAD9	RTC_GPIO9	-	-	-

RTC GPIO Num	GPIO Num	Pad Name	Analog Function			
			1	2	3	4
10	10	TOUCH_PAD10	RTC_GPIO10	-	-	-
11	11	TOUCH_PAD11	RTC_GPIO11	-	-	-
12	12	TOUCH_PAD12	RTC_GPIO12	-	-	-
13	13	TOUCH_PAD13	RTC_GPIO13	-	-	-
14	14	TOUCH_PAD14	RTC_GPIO14	-	-	-
15	15	X32P	RTC_GPIO15	-	-	-
16	16	X32N	RTC_GPIO16	-	-	-
17	17	PDAC1	RTC_GPIO17	-	-	-
18	18	PDAC2	RTC_GPIO18	-	-	-
19	19	RTC_PAD19	RTC_GPIO19	-	-	-
20	20	RTC_PAD20	RTC_GPIO20	-	-	-
21	21	RTC_PAD21	RTC_GPIO21	-	-	-

**Note:** TOUCH\_PAD0 is an internal channel and its analog functions are not lead to a corresponding external GPIO.

### 5.13 Base Address

Users can access GPIO, IO MUX, GPIOSD, Dedicated GPIO, and RTCIO registers with one or two base address(es), which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter 1: [System and Memory](#).

**Table 25: GPIO, IO MUX, GPIOSD, Dedicated GPIO, and RTCIO Base Addresses**

Module	Access to Access Peripheral	Base Address
GPIO	PeriBUS1	0x3F404000
	PeriBUS2	0x60004000
IO MUX	PeriBUS1	0x3F409000
	PeriBUS2	0x60009000
GPIOSD	PeriBUS2	0x60004F00
Deicated GPIO	PeriBUS1	0x3F4CF000
RTCIO	PeriBUS1	0x3F408400
	PeriBUS2	0x60008400

### 5.14 Register Summary

The address in the following part represents the address offset (relative address) with respect to the peripheral base address, not the absolute address. For detailed information about the base address, please refer to Section 5.13.

#### 5.14.1 GPIO Matrix Register Summary

Name	Description	Address	Access
<b>GPIO Configuration Registers</b>			
<a href="#">GPIO_BT_SELECT_REG</a>	GPIO bit selection register	0x0000	R/W

Name	Description	Address	Access
GPIO_OUT_REG	GPIO0 ~ 31 output register	0x0004	R/W
GPIO_OUT_W1TS_REG	GPIO0 ~ 31 output bit set register	0x0008	WO
GPIO_OUT_W1TC_REG	GPIO0 ~ 31 output bit clear register	0x000C	WO
GPIO_OUT1_REG	GPIO32 ~ 53 output register	0x0010	R/W
GPIO_OUT1_W1TS_REG	GPIO32 ~ 53 output bit set register	0x0014	WO
GPIO_OUT1_W1TC_REG	GPIO32 ~ 53 output bit clear register	0x0018	WO
GPIO_SDIO_SELECT_REG	GPIO SDIO selection register	0x001C	R/W
GPIO_ENABLE_REG	GPIO0 ~ 31 output enable register	0x0020	R/W
GPIO_ENABLE_W1TS_REG	GPIO0 ~ 31 output enable bit set register	0x0024	WO
GPIO_ENABLE_W1TC_REG	GPIO0 ~ 31 output enable bit clear register	0x0028	WO
GPIO_ENABLE1_REG	GPIO32 ~ 53 output enable register	0x002C	R/W
GPIO_ENABLE1_W1TS_REG	GPIO32 ~ 53 output enable bit set register	0x0030	WO
GPIO_ENABLE1_W1TC_REG	GPIO32 ~ 53 output enable bit clear register	0x0034	WO
GPIO_STRAP_REG	Bootstrap pin value register	0x0038	RO
GPIO_IN_REG	GPIO0 ~ 31 input register	0x003C	RO
GPIO_IN1_REG	GPIO32 ~ 53 input register	0x0040	RO
GPIO_PIN0_REG	Configuration for GPIO pin 0	0x0074	R/W
GPIO_PIN1_REG	Configuration for GPIO pin 1	0x0078	R/W
GPIO_PIN2_REG	Configuration for GPIO pin 2	0x007C	R/W
...	...	...	...
GPIO_PIN51_REG	Configuration for GPIO pin 51	0x0140	R/W
GPIO_PIN52_REG	Configuration for GPIO pin 52	0x0144	R/W
GPIO_PIN53_REG	Configuration for GPIO pin 53	0x0148	R/W
GPIO_FUNC0_IN_SEL_CFG_REG	Peripheral function 0 input selection register	0x0154	R/W
GPIO_FUNC1_IN_SEL_CFG_REG	Peripheral function 1 input selection register	0x0158	R/W
GPIO_FUNC2_IN_SEL_CFG_REG	Peripheral function 2 input selection register	0x015C	R/W
...	...	...	...
GPIO_FUNC253_IN_SEL_CFG_REG	Peripheral function 253 input selection register	0x0548	R/W
GPIO_FUNC254_IN_SEL_CFG_REG	Peripheral function 254 input selection register	0x054C	R/W
GPIO_FUNC255_IN_SEL_CFG_REG	Peripheral function 255 input selection register	0x0550	R/W
GPIO_FUNC0_OUT_SEL_CFG_REG	Peripheral output selection for GPIO0	0x0554	R/W
GPIO_FUNC1_OUT_SEL_CFG_REG	Peripheral output selection for GPIO1	0x0558	R/W
GPIO_FUNC2_OUT_SEL_CFG_REG	Peripheral output selection for GPIO2	0x055C	R/W
..	...	...	...
GPIO_FUNC51_OUT_SEL_CFG_REG	Peripheral output selection for GPIO51	0x0620	R/W
GPIO_FUNC52_OUT_SEL_CFG_REG	Peripheral output selection for GPIO52	0x0624	R/W
GPIO_FUNC53_OUT_SEL_CFG_REG	Peripheral output selection for GPIO53	0x0628	R/W
GPIO_CLOCK_GATE_REG	GPIO clock gating register	0x062C	R/W
<b>Interrupt Configuration Registers</b>			
GPIO_STATUS_W1TS_REG	GPIO0 ~ 31 interrupt status bit set register	0x0048	WO
GPIO_STATUS_W1TC_REG	GPIO0 ~ 31 interrupt status bit clear register	0x004C	WO
GPIO_STATUS1_W1TS_REG	GPIO32 ~ 53 interrupt status bit set register	0x0054	WO
GPIO_STATUS1_W1TC_REG	GPIO32 ~ 53 interrupt status bit clear register	0x0058	WO

Name	Description	Address	Access
<b>GPIO Interrupt Source Registers</b>			
<a href="#">GPIO_STATUS_NEXT_REG</a>	GPIO0 ~ 31 interrupt source register	0x014C	RO
<a href="#">GPIO_STATUS_NEXT1_REG</a>	GPIO32 ~ 53 interrupt source register	0x0150	RO
<b>Interrupt Status Registers</b>			
<a href="#">GPIO_STATUS_REG</a>	GPIO0 ~ 31 interrupt status register	0x0044	R/W
<a href="#">GPIO_STATUS1_REG</a>	GPIO32 ~ 53 interrupt status register	0x0050	R/W
<a href="#">GPIO_PCPU_INT_REG</a>	GPIO0 ~ 31 PRO_CPU interrupt status register	0x005C	RO
<a href="#">GPIO_PCPU_NMI_INT_REG</a>	GPIO0 ~ 31 PRO_CPU non-maskable interrupt status register	0x0060	RO
<a href="#">GPIO_PCPU_INT1_REG</a>	GPIO32 ~ 53 PRO_CPU interrupt status register	0x0068	RO
<a href="#">GPIO_PCPU_NMI_INT1_REG</a>	GPIO32 ~ 53 PRO_CPU non-maskable interrupt status register	0x006C	RO

### 5.14.2 IO MUX Register Summary

Name	Description	Address	Access
<a href="#">IO_MUX_PIN_CTRL</a>	Clock output configuration register	0x0000	R/W
<a href="#">IO_MUX_GPIO0_REG</a>	Configuration register for pad GPIO0	0x0004	R/W
<a href="#">IO_MUX_GPIO1_REG</a>	Configuration register for pad GPIO1	0x0008	R/W
<a href="#">IO_MUX_GPIO2_REG</a>	Configuration register for pad GPIO2	0x000C	R/W
<a href="#">IO_MUX_GPIO3_REG</a>	Configuration register for pad GPIO3	0x0010	R/W
<a href="#">IO_MUX_GPIO4_REG</a>	Configuration register for pad GPIO4	0x0014	R/W
<a href="#">IO_MUX_GPIO5_REG</a>	Configuration register for pad GPIO5	0x0018	R/W
<a href="#">IO_MUX_GPIO6_REG</a>	Configuration register for pad GPIO6	0x001C	R/W
<a href="#">IO_MUX_GPIO7_REG</a>	Configuration register for pad GPIO7	0x0020	R/W
<a href="#">IO_MUX_GPIO8_REG</a>	Configuration register for pad GPIO8	0x0024	R/W
<a href="#">IO_MUX_GPIO9_REG</a>	Configuration register for pad GPIO9	0x0028	R/W
<a href="#">IO_MUX_GPIO10_REG</a>	Configuration register for pad GPIO10	0x002C	R/W
<a href="#">IO_MUX_GPIO11_REG</a>	Configuration register for pad GPIO11	0x0030	R/W
<a href="#">IO_MUX_GPIO12_REG</a>	Configuration register for pad GPIO12	0x0034	R/W
<a href="#">IO_MUX_GPIO13_REG</a>	Configuration register for pad GPIO13	0x0038	R/W
<a href="#">IO_MUX_GPIO14_REG</a>	Configuration register for pad GPIO14	0x003C	R/W
<a href="#">IO_MUX_XTAL_32K_P_REG</a>	Configuration register for pad XTAL_32K_P	0x0040	R/W
<a href="#">IO_MUX_XTAL_32K_N_REG</a>	Configuration register for pad XTAL_32K_N	0x0044	R/W
<a href="#">IO_MUX_DAC_1_REG</a>	Configuration register for pad DAC_1	0x0048	R/W
<a href="#">IO_MUX_DAC_2_REG</a>	Configuration register for pad DAC_2	0x004C	R/W
<a href="#">IO_MUX_GPIO_19_REG</a>	Configuration register for pad GPIO19	0x0050	R/W
<a href="#">IO_MUX_GPIO_20_REG</a>	Configuration register for pad GPIO20	0x0054	R/W
<a href="#">IO_MUX_GPIO_21_REG</a>	Configuration register for pad GPIO21	0x0058	R/W
<a href="#">IO_MUX_SPICS1_REG</a>	Configuration register for pad SPICS1	0x006C	R/W
<a href="#">IO_MUX_SPIHD_REG</a>	Configuration register for pad SPIHD	0x0070	R/W
<a href="#">IO_MUX_SPIWP_REG</a>	Configuration register for pad SPIWP	0x0074	R/W
<a href="#">IO_MUX_SPICS0_REG</a>	Configuration register for pad SPICS0	0x0078	R/W
<a href="#">IO_MUX_SPICLK_REG</a>	Configuration register for pad SPICLK	0x007C	R/W

Name	Description	Address	Access
<a href="#">IO_MUX_SPIQ_REG</a>	Configuration register for pad SPIQ	0x0080	R/W
<a href="#">IO_MUX_SPID_REG</a>	Configuration register for pad SPID	0x0084	R/W
<a href="#">IO_MUX_GPIO_33_REG</a>	Configuration register for pad GPIO33	0x0088	R/W
<a href="#">IO_MUX_GPIO_34_REG</a>	Configuration register for pad GPIO34	0x008C	R/W
<a href="#">IO_MUX_GPIO_35_REG</a>	Configuration register for pad GPIO35	0x0090	R/W
<a href="#">IO_MUX_GPIO_36_REG</a>	Configuration register for pad GPIO36	0x0094	R/W
<a href="#">IO_MUX_GPIO_37_REG</a>	Configuration register for pad GPIO37	0x0098	R/W
<a href="#">IO_MUX_GPIO_38_REG</a>	Configuration register for pad GPIO38	0x009C	R/W
<a href="#">IO_MUX_MTCK_REG</a>	Configuration register for pad MTCK	0x00A0	R/W
<a href="#">IO_MUX_MTDO_REG</a>	Configuration register for pad MTDO	0x00A4	R/W
<a href="#">IO_MUX_MTDI_REG</a>	Configuration register for pad MTDI	0x00A8	R/W
<a href="#">IO_MUX_MTMS_REG</a>	Configuration register for pad MTMS	0x00AC	R/W
<a href="#">IO_MUX_U0TXD_REG</a>	Configuration register for pad U0TXD	0x00B0	R/W
<a href="#">IO_MUX_U0RXD_REG</a>	Configuration register for pad U0RXD	0x00B4	R/W
<a href="#">IO_MUX_GPIO_45_REG</a>	Configuration register for pad GPIO45	0x00B8	R/W
<a href="#">IO_MUX_GPIO_46_REG</a>	Configuration register for pad GPIO46	0x00BC	R/W

### 5.14.3 Sigma Delta Modulated Output Register Summary

Name	Description	Address	Access
<b>Configuration Registers</b>			
<a href="#">GPIOSD_SIGMADELTA0_REG</a>	Duty Cycle Configure Register of SDM0	0x0000	R/W
<a href="#">GPIOSD_SIGMADELTA1_REG</a>	Duty Cycle Configure Register of SDM1	0x0004	R/W
<a href="#">GPIOSD_SIGMADELTA2_REG</a>	Duty Cycle Configure Register of SDM2	0x0008	R/W
<a href="#">GPIOSD_SIGMADELTA3_REG</a>	Duty Cycle Configure Register of SDM3	0x000C	R/W
<a href="#">GPIOSD_SIGMADELTA4_REG</a>	Duty Cycle Configure Register of SDM4	0x0010	R/W
<a href="#">GPIOSD_SIGMADELTA5_REG</a>	Duty Cycle Configure Register of SDM5	0x0014	R/W
<a href="#">GPIOSD_SIGMADELTA6_REG</a>	Duty Cycle Configure Register of SDM6	0x0018	R/W
<a href="#">GPIOSD_SIGMADELTA7_REG</a>	Duty Cycle Configure Register of SDM7	0x001C	R/W
<a href="#">GPIOSD_SIGMADELTA.CG_REG</a>	Clock Gating Configure Register	0x0020	R/W
<a href="#">GPIOSD_SIGMADELTA_MISC_REG</a>	MISC Register	0x0024	R/W
<a href="#">GPIOSD_SIGMADELTA_VERSION_REG</a>	Version Control Register	0x0028	R/W

### 5.14.4 Dedicated GPIO Register Summary

Name	Description	Address	Access
<b>Configuration registers</b>			
<a href="#">DEDIC_GPIO_OUT_DRT_REG</a>	Dedicated GPIO direct output register	0x0000	WO
<a href="#">DEDIC_GPIO_OUT_MSK_REG</a>	Dedicated GPIO mask output register	0x0004	WO
<a href="#">DEDIC_GPIO_OUT_IDV_REG</a>	Dedicated GPIO individual output register	0x0008	WO
<a href="#">DEDIC_GPIO_OUT_CPU_REG</a>	Dedicated GPIO output mode selection register	0x0010	R/W
<a href="#">DEDIC_GPIO_IN_DLY_REG</a>	Dedicated GPIO input delay configuration register	0x0014	R/W
<a href="#">DEDIC_GPIO_INTR_RCGN_REG</a>	Dedicated GPIO interrupts generation mode register	0x001C	R/W
<b>Status registers</b>			



Name	Description	Address	Access
<a href="#">DEDIC_GPIO_OUT_SCAN_REG</a>	Dedicated GPIO output status register	0x000C	RO
<a href="#">DEDIC_GPIO_IN_SCAN_REG</a>	Dedicated GPIO input status register	0x0018	RO
<b>Interrupt registers</b>			
<a href="#">DEDIC_GPIO_INTR_RAW_REG</a>	Raw interrupt status	0x0020	RO
<a href="#">DEDIC_GPIO_INTR_RLS_REG</a>	Interrupt enable bits	0x0024	R/W
<a href="#">DEDIC_GPIO_INTR_ST_REG</a>	Masked interrupt status	0x0028	RO
<a href="#">DEDIC_GPIO_INTR_CLR_REG</a>	Interrupt clear bits	0x002C	WO

### 5.14.5 RTC IO MUX Register Summary

Name	Description	Address	Access
<b>GPIO Configuration and Data Registers</b>			
<a href="#">RTCIO_RTC_GPIO_OUT_REG</a>	RTC GPIO output register	0x0000	R/W
<a href="#">RTCIO_RTC_GPIO_OUT_W1TS_REG</a>	RTC GPIO output bit set register	0x0004	WO
<a href="#">RTCIO_RTC_GPIO_OUT_W1TC_REG</a>	RTC GPIO output bit clear register	0x0008	WO
<a href="#">RTCIO_RTC_GPIO_ENABLE_REG</a>	RTC GPIO output enable register	0x000C	R/W
<a href="#">RTCIO_RTC_GPIO_ENABLE_W1TS_REG</a>	RTC GPIO output enable bit set register	0x0010	WO
<a href="#">RTCIO_RTC_GPIO_ENABLE_W1TC_REG</a>	RTC GPIO output enable bit clear register	0x0014	WO
<a href="#">RTCIO_RTC_GPIO_STATUS_REG</a>	RTC GPIO interrupt status register	0x0018	R/W
<a href="#">RTCIO_RTC_GPIO_STATUS_W1TS_REG</a>	RTC GPIO interrupt status bit set register	0x001C	WO
<a href="#">RTCIO_RTC_GPIO_STATUS_W1TC_REG</a>	RTC GPIO interrupt status bit clear register	0x0020	WO
<a href="#">RTCIO_RTC_GPIO_IN_REG</a>	RTC GPIO input register	0x0024	RO
<a href="#">RTCIO_RTC_GPIO_PIN0_REG</a>	RTC configuration for pin 0	0x0028	R/W
<a href="#">RTCIO_RTC_GPIO_PIN1_REG</a>	RTC configuration for pin 1	0x002C	R/W
<a href="#">RTCIO_RTC_GPIO_PIN2_REG</a>	RTC configuration for pin 2	0x0030	R/W
<a href="#">RTCIO_RTC_GPIO_PIN3_REG</a>	RTC configuration for pin 3	0x0034	R/W
...	...	...	...
<a href="#">RTCIO_RTC_GPIO_PIN19_REG</a>	RTC configuration for pin 19	0x0074	R/W
<a href="#">RTCIO_RTC_GPIO_PIN20_REG</a>	RTC configuration for pin 20	0x0078	R/W
<a href="#">RTCIO_RTC_GPIO_PIN21_REG</a>	RTC configuration for pin 21	0x007C	R/W
<b>GPIO RTC Function Configuration Registers</b>			
<a href="#">RTCIO_TOUCH_PAD0_REG</a>	Touch pad 0 configuration register	0x0084	R/W
<a href="#">RTCIO_TOUCH_PAD1_REG</a>	Touch pad 1 configuration register	0x0088	R/W
<a href="#">RTCIO_TOUCH_PAD2_REG</a>	Touch pad 2 configuration register	0x008C	R/W
...	...	...	...
<a href="#">RTCIO_TOUCH_PAD13_REG</a>	Touch pad 13 configuration register	0x00B8	R/W
<a href="#">RTCIO_TOUCH_PAD14_REG</a>	Touch pad 14 configuration register	0x00BC	R/W
<a href="#">RTCIO_XTAL_32P_PAD_REG</a>	32KHz crystal P-pad configuration register	0x00C0	R/W
<a href="#">RTCIO_XTAL_32N_PAD_REG</a>	32KHz crystal N-pad configuration register	0x00C4	R/W
<a href="#">RTCIO_PAD_DAC1_REG</a>	DAC1 configuration register	0x00C8	R/W
<a href="#">RTCIO_PAD_DAC2_REG</a>	DAC2 configuration register	0x00CC	R/W
<a href="#">RTCIO_RTC_PAD19_REG</a>	Touch pad 19 configuration register	0x00D0	R/W
<a href="#">RTCIO_RTC_PAD20_REG</a>	Touch pad 20 configuration register	0x00D4	R/W
<a href="#">RTCIO_RTC_PAD21_REG</a>	Touch pad 21 configuration register	0x00D8	R/W

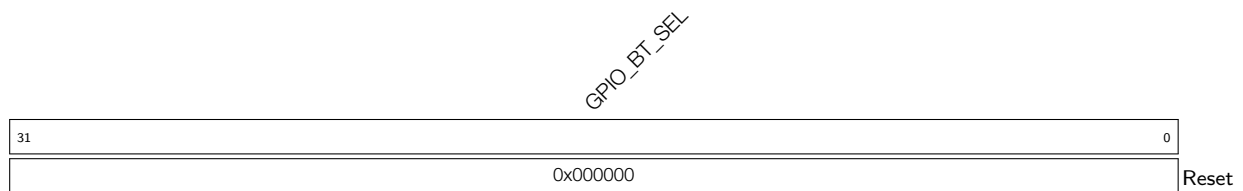
Name	Description	Address	Access
<a href="#">RTCIO_XTL_EXT_CTR_REG</a>	Crystal power down enable GPIO source	0x00E0	R/W
<a href="#">RTCIO_SAR_I2C_IO_REG</a>	RTC I2C pad selection	0x00E4	R/W

## 5.15 Registers

The address in the following part represents the address offset (relative address) with respect to the peripheral base address, not the absolute address. For detailed information about the base address, please refer to Section 5.13.

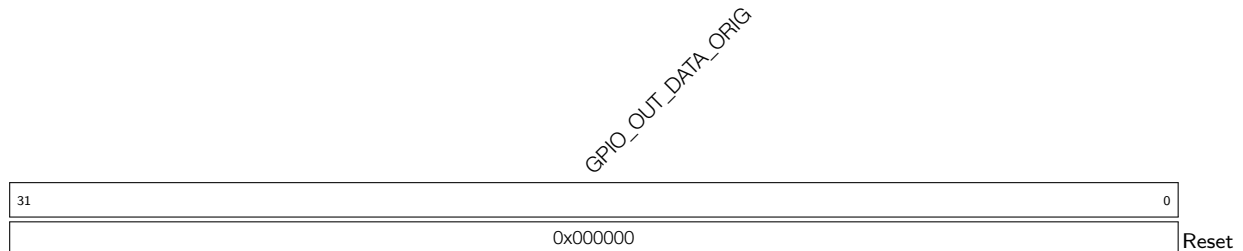
### 5.15.1 GPIO Matrix Registers

**Register 5.1: GPIO\_BT\_SELECT\_REG (0x0000)**



**GPIO\_BT\_SEL** Reserved (R/W)

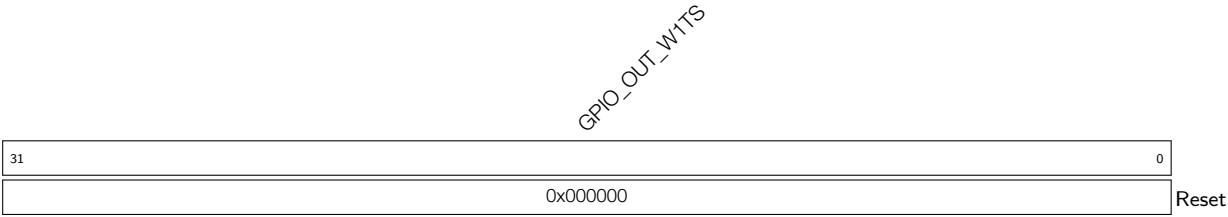
**Register 5.2: GPIO\_OUT\_REG (0x0004)**



**GPIO\_OUT\_DATA\_ORIG** GPIO0 ~ 31 output value in simple GPIO output mode. The values of bit0 ~ bit31 correspond to the output value of GPIO0 ~ GPIO31 respectively. Bit22 ~ bit25 are invalid. (R/W)

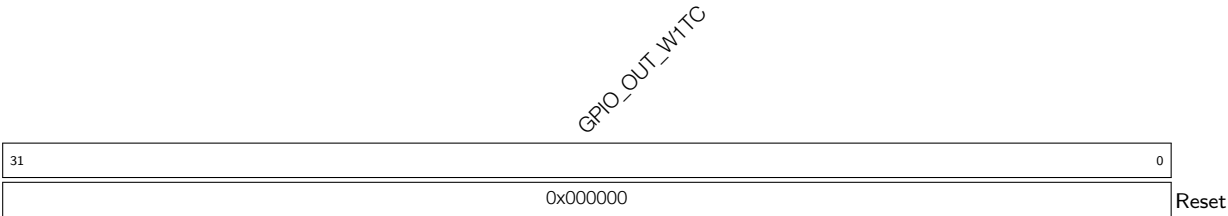


Register 5.3: GPIO\_OUT\_W1TS\_REG (0x0008)



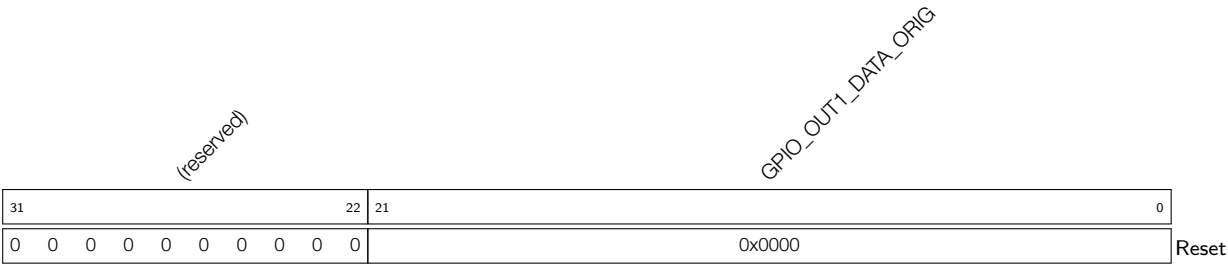
**GPIO\_OUT\_W1TS** GPIO00 ~ 31 output set register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_OUT\\_REG](#) will be set to 1. Recommended operation: use this register to set [GPIO\\_OUT\\_REG](#). (WO)

Register 5.4: GPIO\_OUT\_W1TC\_REG (0x000C)



**GPIO\_OUT\_W1TC** GPIO00 ~ 31 output clear register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_OUT\\_REG](#) will be cleared. Recommended operation: use this register to clear [GPIO\\_OUT\\_REG](#). (WO)

Register 5.5: GPIO\_OUT1\_REG (0x0010)



**GPIO\_OUT1\_DATA\_ORIG** GPIO32 ~ 53 output value in simple GPIO output mode. The values of bit0 ~ bit13 correspond to GPIO32 ~ GPIO45. Bit14 ~ bit21 are invalid. (R/W)

Register 5.6: GPIO\_OUT1\_W1TS\_REG (0x0014)

(reserved)										GPIO_OUT1_W1TS												
31										22	21											0
0	0	0	0	0	0	0	0	0	0		0x0000										Reset	

**GPIO\_OUT1\_W1TS** GPIO32 ~ 53 output value set register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_OUT1\\_REG](#) will be set to 1. Recommended operation: use this register to set [GPIO\\_OUT1\\_REG](#). (WO)

Register 5.7: GPIO\_OUT1\_W1TC\_REG (0x0018)

(reserved)										GPIO_OUT1_W1TC																						
31											22	0																				
0	0	0	0	0	0	0	0	0	0	0	0x0000																					Reset

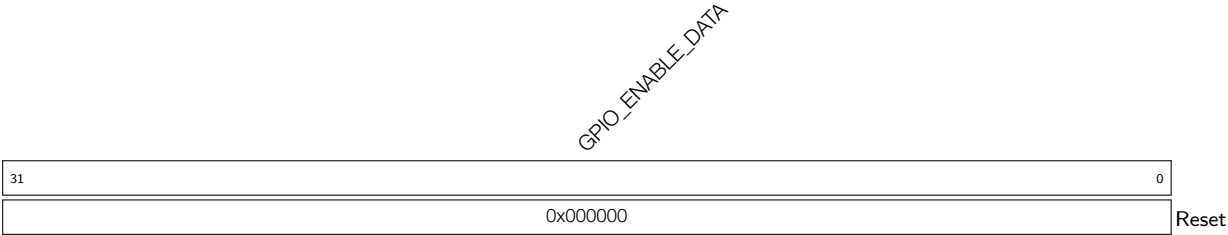
**GPIO\_OUT1\_W1TC** GPIO32 ~ 53 output value clear register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_OUT1\\_REG](#) will be cleared. Recommended operation: use this register to clear [GPIO\\_OUT1\\_REG](#). (WO)

Register 5.8: GPIO\_SDIO\_SELECT\_REG (0x001C)

(reserved)																GPIO_SDIO_SEL						
31																8	7					0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0				Reset

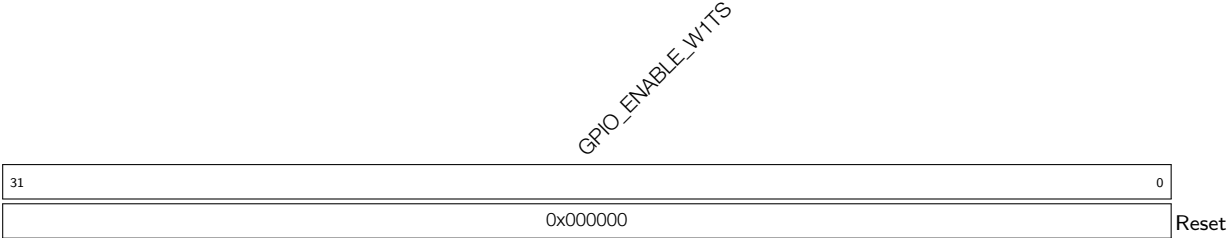
**GPIO\_SDIO\_SEL** Reserved (R/W)

Register 5.9: GPIO\_ENABLE\_REG (0x0020)



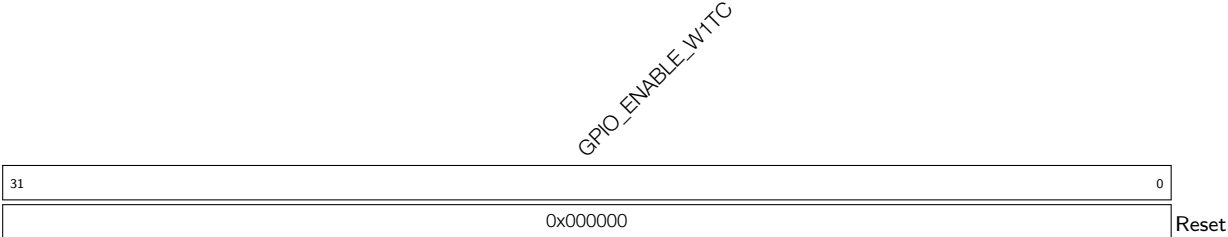
**GPIO\_ENABLE\_DATA** GPIO0 ~ 31 output enable register. (R/W)

Register 5.10: GPIO\_ENABLE\_W1TS\_REG (0x0024)



**GPIO\_ENABLE\_W1TS** GPIO0 ~ 31 output enable set register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_ENABLE\\_REG](#) will be set to 1. Recommended operation: use this register to set [GPIO\\_ENABLE\\_REG](#). (WO)

Register 5.11: GPIO\_ENABLE\_W1TC\_REG (0x0028)



**GPIO\_ENABLE\_W1TC** GPIO0 ~ 31 output enable clear register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_ENABLE\\_REG](#) will be cleared. Recommended operation: use this register to clear [GPIO\\_ENABLE\\_REG](#). (WO)

**Register 5.12: GPIO\_ENABLE1\_REG (0x002C)**

(reserved)										GPIO_ENABLE1_DATA																																
31										22																						21										0
0 0 0 0 0 0 0 0 0 0										0x0000																						Reset										

**GPIO\_ENABLE1\_DATA** GPIO32 ~ 53 output enable register. (R/W)

**Register 5.13: GPIO\_ENABLE1\_W1TS\_REG (0x0030)**

(reserved)										GPIO_ENABLE1_W1TS																																
31										22										21																						0
0 0 0 0 0 0 0 0 0 0										0x0000																						Reset										

**GPIO\_ENABLE1\_W1TS** GPIO32 ~ 53 output enable set register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_ENABLE1\\_REG](#) will be set to 1. Recommended operation: use this register to set [GPIO\\_ENABLE1\\_REG](#). (WO)

**Register 5.14: GPIO\_ENABLE1\_W1TC\_REG (0x0034)**

(reserved)										GPIO_ENABLE1_W1TC																															
31										22										21																					0
0 0 0 0 0 0 0 0 0 0										0x0000																					Reset										

**GPIO\_ENABLE1\_W1TC** GPIO32 ~ 53 output enable clear register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_ENABLE1\\_REG](#) will be cleared. Recommended operation: use this register to clear [GPIO\\_ENABLE1\\_REG](#). (WO)

**Register 5.15: GPIO\_STRAP\_REG (0x0038)**

(reserved)																GPIO_STRAPPING																
31															16	15															0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
																																Reset

**GPIO\_STRAPPING** GPIO strapping values: bit4 ~ bit2 correspond to stripping pins GPIO45, GPIO0, and GPIO46 respectively. (RO)

**Register 5.16: GPIO\_IN\_REG (0x003C)**

GPIO_IN_DATA_NEXT																																
31																															0	
																																0
																																Reset

**GPIO\_IN\_DATA\_NEXT** GPIO0 ~ 31 input value. Each bit represents a pad input value, 1 for high level and 0 for low level. (RO)

**Register 5.17: GPIO\_IN1\_REG (0x0040)**

(reserved)											GPIO_IN_DATA1_NEXT																				
31											22	21																			0
0	0	0	0	0	0	0	0	0	0	0	0																			Reset	

**GPIO\_IN\_DATA1\_NEXT** GPIO32 ~ 53 input value. Each bit represents a pad input value. (RO)

**Register 5.18: GPIO\_PIN $n$ \_REG ( $n$ : 0-53) (0x0074+4\* $n$ )**

(reserved)																GPIO_PIN <sub>n</sub> _INT_ENA				GPIO_PIN <sub>n</sub> _CONFIG				GPIO_PIN <sub>n</sub> _WAKEUP_ENABLE				(reserved)				GPIO_PIN <sub>n</sub> _SYNC1_BYPASS				GPIO_PIN <sub>n</sub> _PAD_DRIVER				GPIO_PIN <sub>n</sub> _SYNC2_BYPASS			
31																	18	17			13	12	11	10	9			7	6	5	4	3	2	1	0								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0	0x0	0	0x0	0	0	0x0	0	0x0	Reset																		

**GPIO\_PIN $n$ \_SYNC2\_BYPASS** For the second stage synchronization, GPIO input data can be synchronized on either edge of the APB clock. 0: no synchronization; 1: synchronized on falling edge; 2 and 3: synchronized on rising edge. (R/W)

**GPIO\_PIN $n$ \_PAD\_DRIVER** Pad driver selection. 0: normal output; 1: open drain output. (R/W)

**GPIO\_PIN $n$ \_SYNC1\_BYPASS** For the first stage synchronization, GPIO input data can be synchronized on either edge of the APB clock. 0: no synchronization; 1: synchronized on falling edge; 2 and 3: synchronized on rising edge. (R/W)

**GPIO\_PIN $n$ \_INT\_TYPE** Interrupt type selection. 0: GPIO interrupt disabled; 1: rising edge trigger; 2: falling edge trigger; 3: any edge trigger; 4: low level trigger; 5: high level trigger. (R/W)

**GPIO\_PIN $n$ \_WAKEUP\_ENABLE** GPIO wake-up enable bit, only wakes up the CPU from Light-sleep (R/W)

**GPIO\_PIN $n$ \_CONFIG** Reserved (R/W)

**GPIO\_PIN $n$ \_INT\_ENA** Interrupt enable bits. bit13: CPU interrupt enabled; bit14: CPU non-maskable interrupt enabled. (R/W)

**Register 5.19: GPIO\_FUNC $n$ \_IN\_SEL\_CFG\_REG ( $n$ : 0-255) (0x0154+4\* $n$ )**

(reserved)																								GPIO_SIG <sub>n</sub> _IN_SEL				GPIO_FUNC <sub>n</sub> _IN_INV_SEL				GPIO_FUNC <sub>n</sub> _IN_SEL			
31																									8	7	6	5	0						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0	Reset			

**GPIO\_FUNC $n$ \_IN\_SEL** Selection control for peripheral input signal  $m$ , selects a pad from the 54 GPIO matrix pads to connect this input signal. Or selects 0x38 for a constantly high input or 0x3C for a constantly low input. (R/W)

**GPIO\_FUNC $n$ \_IN\_INV\_SEL** Invert the input value. 1: invert enabled; 0: invert disabled. (R/W)

**GPIO\_SIG $n$ \_IN\_SEL** Bypass GPIO matrix. 1: route signals via GPIO matrix, 0: connect signals directly to peripheral configured in IO\_MUX. (R/W)

**Register 5.20: GPIO\_FUNC $n$ \_OUT\_SEL\_CFG\_REG ( $n$ : 0-53) (0x0554+4\* $n$ )**

(reserved)																								GPIO_FUNC <sub>n</sub> _OEN_INV_SEL			GPIO_FUNC <sub>n</sub> _OEN_SEL			GPIO_FUNC <sub>n</sub> _OUT_INV_SEL			GPIO_FUNC <sub>n</sub> _OUT_SEL													
31																								12	11	10	9	8	0																	
0 0																								0	0	0	0x100																		Reset	

**GPIO\_FUNC $n$ \_OUT\_SEL** Selection control for GPIO output  $n$ . If a value  $s$  ( $0 \leq s < 256$ ) is written to this field, the peripheral output signal  $s$  will be connected to GPIO output  $n$ . If a value 256 is written to this field, bit  $n$  of [GPIO\\_OUT\\_REG](#)/[GPIO\\_OUT1\\_REG](#) and [GPIO\\_ENABLE\\_REG](#)/[GPIO\\_ENABLE1\\_REG](#) will be selected as the output value and output enable. (R/W)

**GPIO\_FUNC $n$ \_OUT\_INV\_SEL** 0: Do not invert the output value; 1: Invert the output value. (R/W)

**GPIO\_FUNC $n$ \_OEN\_SEL** 0: Use output enable signal from peripheral; 1: Force the output enable signal to be sourced from bit  $n$  of [GPIO\\_ENABLE\\_REG](#). (R/W)

**GPIO\_FUNC $n$ \_OEN\_INV\_SEL** 0: Do not invert the output enable signal; 1: Invert the output enable signal. (R/W)

**Register 5.21: GPIO\_CLOCK\_GATE\_REG (0x062C)**

(reserved)																																GPIO_CLK_EN		
31																															1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Reset

**GPIO\_CLK\_EN** Clock gating enable bit. If set to 1, the clock is free running. (R/W)

**Register 5.22: GPIO\_STATUS\_W1TS\_REG (0x0048)**

GPIO_STATUS_W1TS																															
31																															0
0x000000																															
Reset																															

**GPIO\_STATUS\_W1TS** GPIO0 ~ 31 interrupt status set register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_STATUS\\_INTERRUPT](#) will be set to 1. Recommended operation: use this register to set [GPIO\\_STATUS\\_INTERRUPT](#). (WO)

**Register 5.23: GPIO\_STATUS\_W1TC\_REG (0x004C)**

GPIO_STATUS_W1TC																															
0x000000																															
Reset																															

**GPIO\_STATUS\_W1TC** GPIO0 ~ 31 interrupt status clear register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_STATUS\\_INTERRUPT](#) will be cleared. Recommended operation: use this register to clear [GPIO\\_STATUS\\_INTERRUPT](#). (WO)

**Register 5.24: GPIO\_STATUS1\_W1TS\_REG (0x0054)**

(reserved)										GPIO_STATUS1_W1TS																				
31											22	21																	0	
0	0	0	0	0	0	0	0	0	0	0	0x0000																			
Reset																														

**GPIO\_STATUS1\_W1TS** GPIO32 ~ 53 interrupt status set register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_STATUS1\\_REG](#) will be set to 1. Recommended operation: use this register to set [GPIO\\_STATUS1\\_REG](#). (WO)

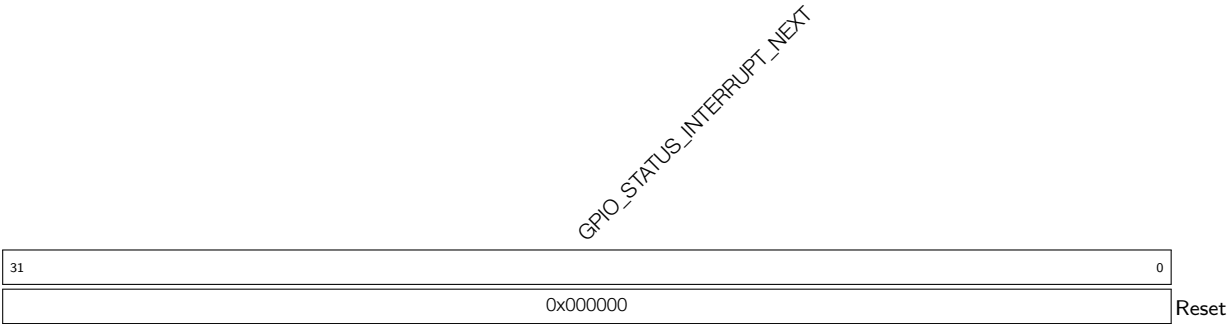
**Register 5.25: GPIO\_STATUS1\_W1TC\_REG (0x0058)**

(reserved)										GPIO_STATUS1_W1TC																																																																															
31										22																						21																				0																																					
0										0										0										0										0										0										0										0x0000										Reset									

**GPIO\_STATUS1\_W1TC** GPIO32 ~ 53 interrupt status clear register. If the value 1 is written to a bit here, the corresponding bit in [GPIO\\_STATUS1\\_REG](#) will be cleared. Recommended operation: use this register to clear [GPIO\\_STATUS1\\_REG](#). (WO)

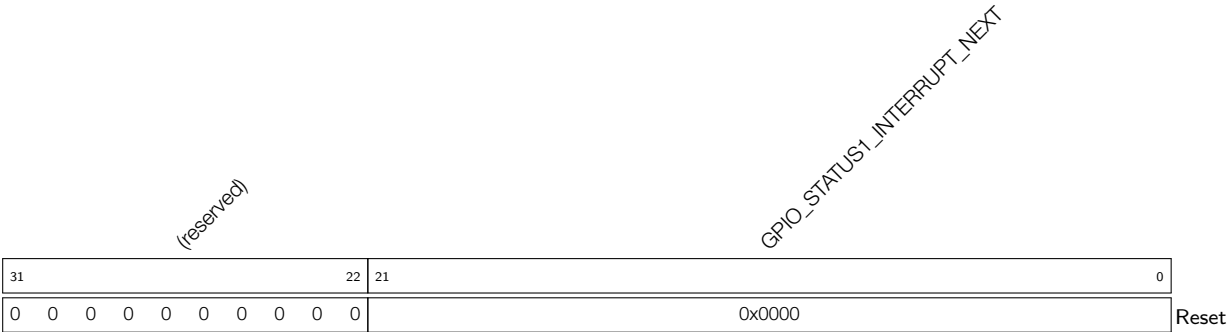


Register 5.26: GPIO\_STATUS\_NEXT\_REG (0x014C)



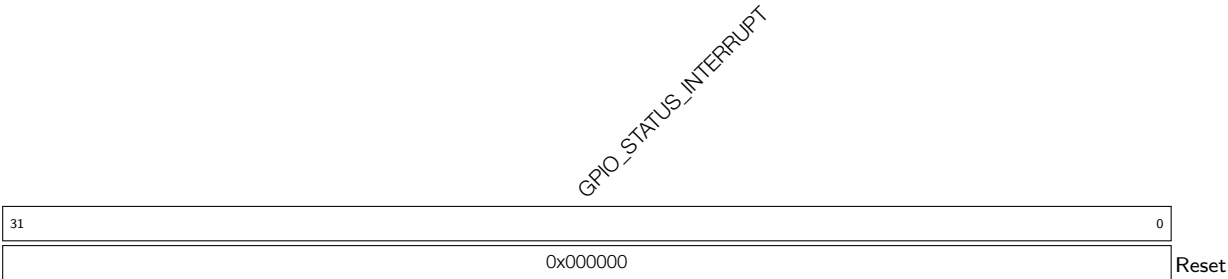
**GPIO\_STATUS\_INTERRUPT\_NEXT** Interrupt source signal of GPIO0 ~ 31, could be rising edge interrupt, falling edge interrupt, level sensitive interrupt and any edge interrupt. (RO)

Register 5.27: GPIO\_STATUS\_NEXT1\_REG (0x0150)



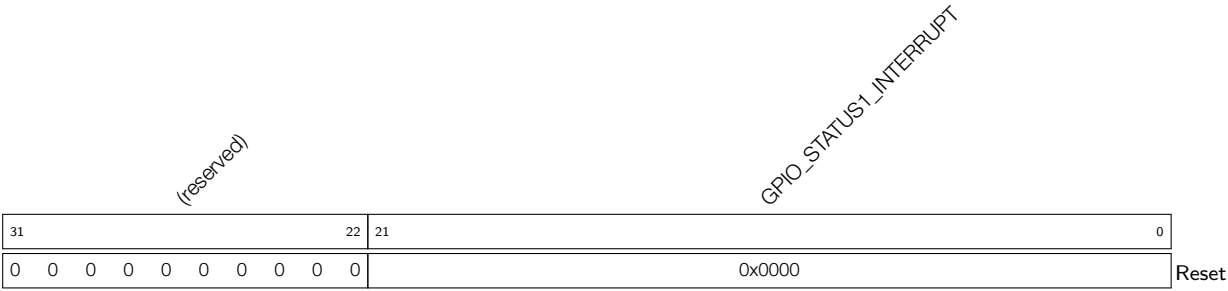
**GPIO\_STATUS1\_INTERRUPT\_NEXT** Interrupt source signal of GPIO32 ~ 53. (RO)

Register 5.28: GPIO\_STATUS\_REG (0x0044)



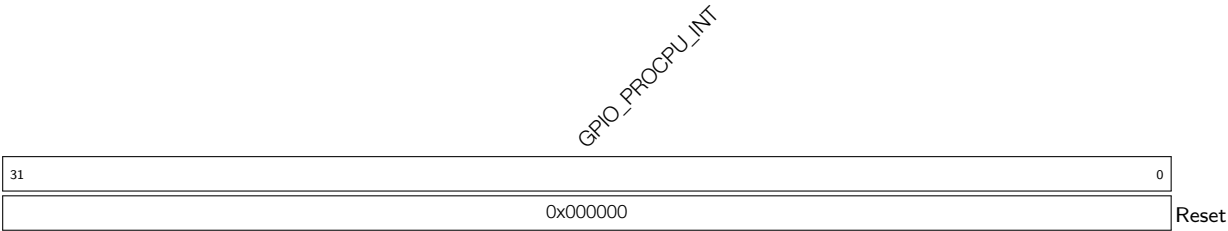
**GPIO\_STATUS\_INTERRUPT** GPIO0 ~ 31 interrupt status register. (R/W)

Register 5.29: GPIO\_STATUS1\_REG (0x0050)



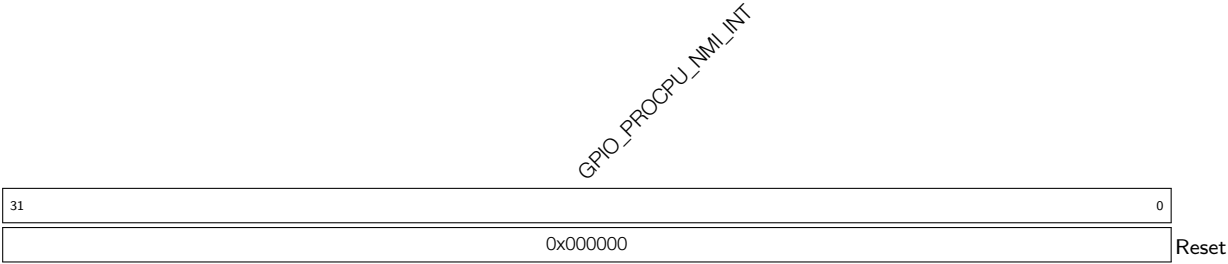
**GPIO\_STATUS1\_INTERRUPT** GPIO32 ~ 53 interrupt status register. (R/W)

Register 5.30: GPIO\_PCPU\_INT\_REG (0x005C)



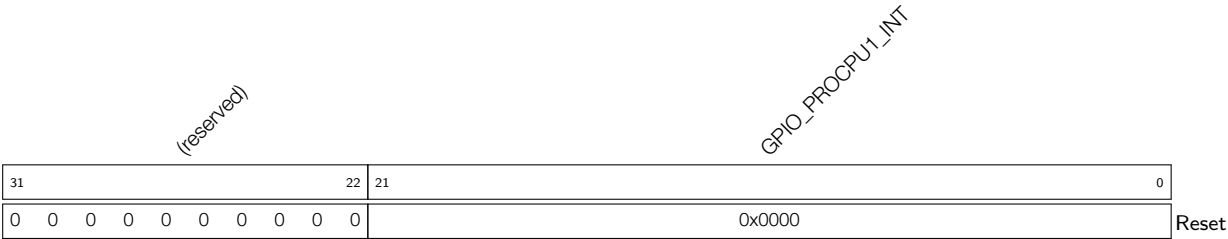
**GPIO\_PROCPU\_INT** GPIO0 ~ 31 PRO\_CPU interrupt status. This interrupt status is corresponding to the bit in [GPIO\\_STATUS\\_REG](#) when assert (high) enable signal (bit13 of [GPIO\\_PIN<sub>n</sub>\\_REG](#)). (RO)

Register 5.31: GPIO\_PCPU\_NMI\_INT\_REG (0x0060)



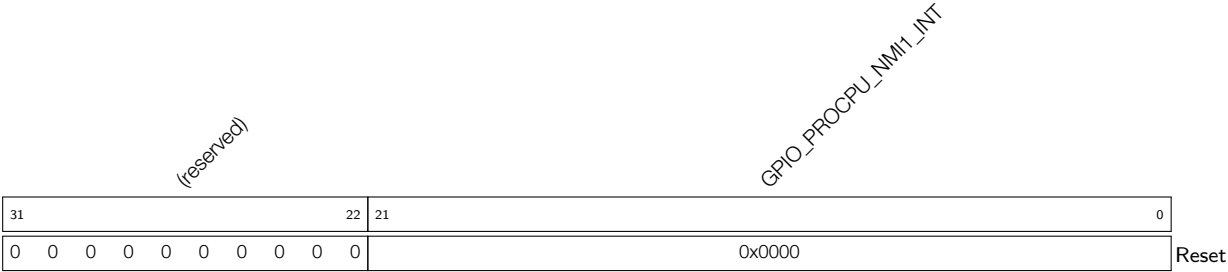
**GPIO\_PROCPU\_NMI\_INT** GPIO0 ~ 31 PRO\_CPU non-maskable interrupt status. This interrupt status is corresponding to the bit in [GPIO\\_STATUS\\_REG](#) when assert (high) enable signal (bit 14 of [GPIO\\_PIN<sub>n</sub>\\_REG](#)). (RO)

Register 5.32: GPIO\_PCPU\_INT1\_REG (0x0068)



**GPIO\_PROCPU1\_INT** GPIO32 ~ 53 PRO\_CPU interrupt status. This interrupt status is corresponding to the bit in [GPIO\\_STATUS1\\_REG](#) when assert (high) enable signal (bit 13 of [GPIO\\_PINn\\_REG](#)). (RO)

Register 5.33: GPIO\_PCPU\_NMI\_INT1\_REG (0x006C)



**GPIO\_PROCPU\_NMI1\_INT** GPIO32 ~ 53 PRO\_CPU non-maskable interrupt status. This interrupt status is corresponding to bit in [GPIO\\_STATUS1\\_REG](#) when assert (high) enable signal (bit 14 of [GPIO\\_PINn\\_REG](#)). (RO)



**Register 5.35: IO\_MUX\_***n***\_REG (*n*: GPIO0-GPIO21, GPIO26-GPIO46) (0x0010+4\**n*)**

(reserved)																IO_MUX_FILTER_EN		IO_MUX_MCU_SEL		IO_MUX_FUN_DRV		IO_MUX_FUN_IE		IO_MUX_FUN_WPU		(reserved)		IO_MUX_MCU_IE		IO_MUX_MCU_WPU		IO_MUX_MCU_WPD		IO_MUX_SLP_SEL		IO_MUX_MCU_OE			
31																16	15	14	12	11	10	9	8	7	6	5	4	3	2	1	0								
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x0	0x0	0x2	0	0	0	00	0	0	0	0	0	0	0	0	Reset								

**IO\_MUX\_MCU\_OE** Output enable of the pad in sleep mode. 1: Output enabled; 0: Output disabled.  
(R/W)

**IO\_MUX\_SLP\_SEL** Sleep mode selection of this pad. Set to 1 to put the pad in sleep mode. (R/W)

**IO\_MUX\_MCU\_WPD** Pull-down enable of the pad during sleep mode. 1: Internal pull-down enabled; 0: Internal pull-down disabled. (R/W)

**IO\_MUX\_MCU\_WPU** Pull-up enable of the pad during sleep mode. 1: Internal pull-up enabled; 0: Internal pull-up disabled.

**IO\_MUX\_MCU\_IE** Input enable of the pad during sleep mode. 1: Input enabled; 0: Input disabled.  
(R/W)

**IO\_MUX\_FUN\_WPD** Pull-down enable of the pad. 1: Pull-down enabled; 0: Pull-down disabled.  
(R/W)

**IO\_MUX\_FUN\_WPU** Pull-up enable of the pad. 1: Internal pull-up enabled; 0: Internal pull-up disabled. (R/W)

**IO\_MUX\_FUN\_IE** Input enable of the pad. 1: Input enabled; 0: Input disabled. (R/W)

**IO\_MUX\_FUN\_DRV** Select the drive strength of the pad. 0: ~5 mA; 1: ~10 mA; 2: ~20 mA; 3: ~40 mA. (R/W)

**IO\_MUX\_MCU\_SEL** Select IO MUX function for this signal. 0: Select Function 0; 1: Select Function 1, etc. (R/W)

**IO\_MUX\_FILTER\_EN** Enable filter for pin input signals. 1: Filter enabled; 2: Filter disabled. (R/W)



**Register 5.39: GPIOSD\_SIGMADELTA\_VERSION\_REG (0x0028)**

(reserved)				GPIOSD_GPIO_SD_DATE																											
31	28	27	0																												
0	0	0	0	0x1802260																											Reset

Reset

**GPIOSD\_GPIO\_SD\_DATE** Version control register. (R/W)

### 5.15.4 Dedicated GPIO Registers

**Register 5.40: DEDIC\_GPIO\_OUT\_DRT\_REG (0x0000)**

(reserved)																												DEDIC_GPIO_OUT_DRT_VLAUE																							
31																												8								7								0							
0 0																												0x0								Reset															

Reset

**DEDIC\_GPIO\_OUT\_DRT\_VLAUE** This register is used to configure direct output value of 8-channel dedicated GPIO. (WO)**Register 5.41: DEDIC\_GPIO\_OUT\_MSK\_REG (0x0004)**

(reserved)																DEDIC_GPIO_OUT_MSK								DEDIC_GPIO_OUT_VALUE										
31																16	15								8	7								0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0								0x0								Reset		

Reset

**DEDIC\_GPIO\_OUT\_VALUE** This register is used to configure updated output value of 8-channel dedicated GPIO. (WO)**DEDIC\_GPIO\_OUT\_MSK** This register is used to configure channels which would be updated. 1: corresponding channel's output would be updated. (WO)

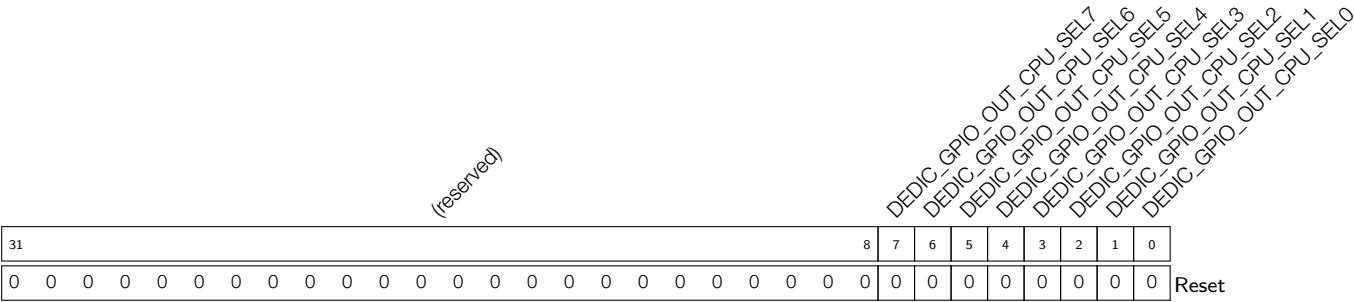
## 128

ESP32-S2 TRM (Preliminary V0.5)

**DEDIC\_GPIO\_OUT\_IDV\_CH7** Configure channel 7 output value. 0: hold output value. 1: set output value. 2: clear output value. 3: inverse output value. (WO)



Register 5.43: DEDIC\_GPIO\_OUT\_CPU\_REG (0x0010)



- DEDIC\_GPIO\_OUT\_CPU\_SEL0 Select GPIO out value configured by registers or CPU instructions for channel 0. 0: Configured by registers. 1: Configured by CPU instructions. (R/W)
- DEDIC\_GPIO\_OUT\_CPU\_SEL1 Select GPIO out value configured by registers or CPU instructions for channel 1. 0: Configured by registers. 1: Configured by CPU instructions. (R/W)
- DEDIC\_GPIO\_OUT\_CPU\_SEL2 Select GPIO out value configured by registers or CPU instructions for channel 2. 0: Configured by registers. 1: Configured by CPU instructions. (R/W)
- DEDIC\_GPIO\_OUT\_CPU\_SEL3 Select GPIO out value configured by registers or CPU instructions for channel 3. 0: Configured by registers. 1: Configured by CPU instructions. (R/W)
- DEDIC\_GPIO\_OUT\_CPU\_SEL4 Select GPIO out value configured by registers or CPU instructions for channel 4. 0: Configured by registers. 1: Configured by CPU instructions. (R/W)
- DEDIC\_GPIO\_OUT\_CPU\_SEL5 Select GPIO out value configured by registers or CPU instructions for channel 5. 0: Configured by registers. 1: Configured by CPU instructions. (R/W)
- DEDIC\_GPIO\_OUT\_CPU\_SEL6 Select GPIO out value configured by registers or CPU instructions for channel 6. 0: Configured by registers. 1: Configured by CPU instructions. (R/W)
- DEDIC\_GPIO\_OUT\_CPU\_SEL7 Select GPIO out value configured by registers or CPU instructions for channel 7. 0: Configured by registers. 1: Configured by CPU instructions. (R/W)

[Submit Documentation Feedback](#)

ESP32-S2 TRM (Preliminary V0.5)

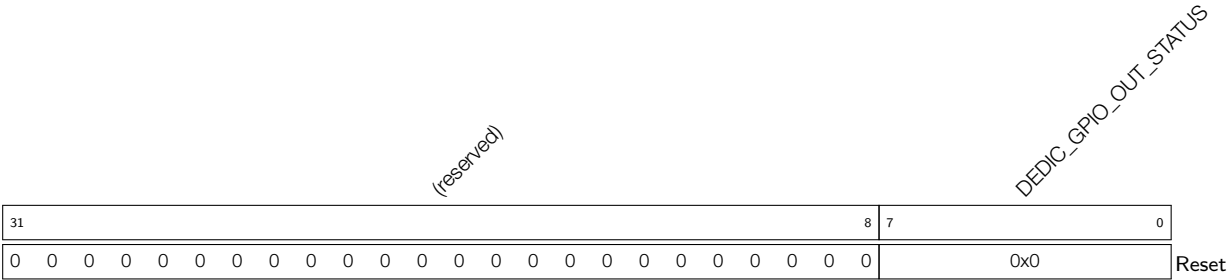
**DEDIC\_GPIO\_IN\_DLY\_CH7** Configure GPIO7 input delay. 0: no delay. 1: one clock delay. 2: two clock delay. 3: three clock delay. (R/W)

## 131

ESP32-S2 TRM (Preliminary V0.5)

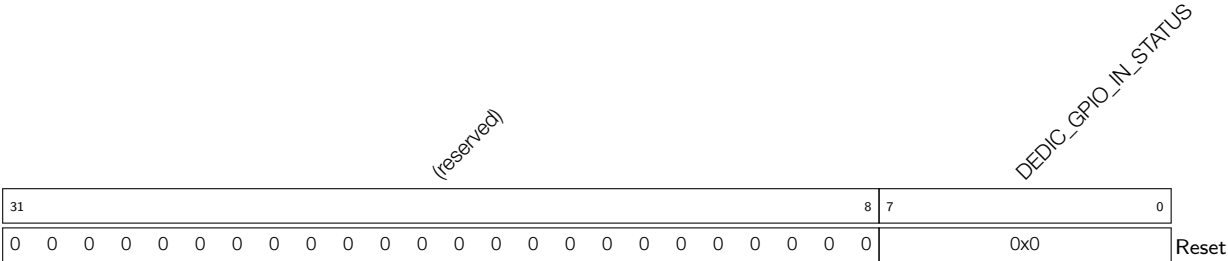
**DEDIC\_GPIO\_INTR\_MODE\_CH7** Configure channel 7 interrupt generate mode. 0/1: do not generate interrupt. 2: low level trigger. 3: high level trigger. 4: falling edge trigger. 5: raising edge trigger. 6/7: falling and raising edge trigger. (R/W)

Register 5.46: DEDIC\_GPIO\_OUT\_SCAN\_REG (0x000C)



**DEDIC\_GPIO\_OUT\_STATUS** GPIO output value configured by [DEDIC\\_GPIO\\_OUT\\_DRT\\_REG](#), [DEDIC\\_GPIO\\_OUT\\_MSK\\_REG](#), and [DEDIC\\_GPIO\\_OUT\\_IDV\\_REG](#). (RO)

Register 5.47: DEDIC\_GPIO\_IN\_SCAN\_REG (0x0018)



**DEDIC\_GPIO\_IN\_STATUS** GPIO input value after configured by [DEDIC\\_GPIO\\_IN\\_DLY\\_REG](#). (RO)

**Register 5.48: DEDIC\_GPIO\_INTR\_RAW\_REG (0x0020)**

(reserved)																								DEDIC_GPIO7_INT_RAW DEDIC_GPIO6_INT_RAW DEDIC_GPIO5_INT_RAW DEDIC_GPIO4_INT_RAW DEDIC_GPIO3_INT_RAW DEDIC_GPIO2_INT_RAW DEDIC_GPIO1_INT_RAW DEDIC_GPIO0_INT_RAW									
31																								8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset		

**DEDIC\_GPIO0\_INT\_RAW** This interrupt raw bit turns to high level when dedicated GPIO0 has level/edge change configured by [DEDIC\\_GPIO\\_INTR\\_RCGN\\_REG](#). (RO)

**DEDIC\_GPIO1\_INT\_RAW** This interrupt raw bit turns to high level when dedicated GPIO1 has level/edge change configured by [DEDIC\\_GPIO\\_INTR\\_RCGN\\_REG](#). (RO)

**DEDIC\_GPIO2\_INT\_RAW** This interrupt raw bit turns to high level when dedicated GPIO2 has level/edge change configured by [DEDIC\\_GPIO\\_INTR\\_RCGN\\_REG](#). (RO)

**DEDIC\_GPIO3\_INT\_RAW** This interrupt raw bit turns to high level when dedicated GPIO3 has level/edge change configured by [DEDIC\\_GPIO\\_INTR\\_RCGN\\_REG](#). (RO)

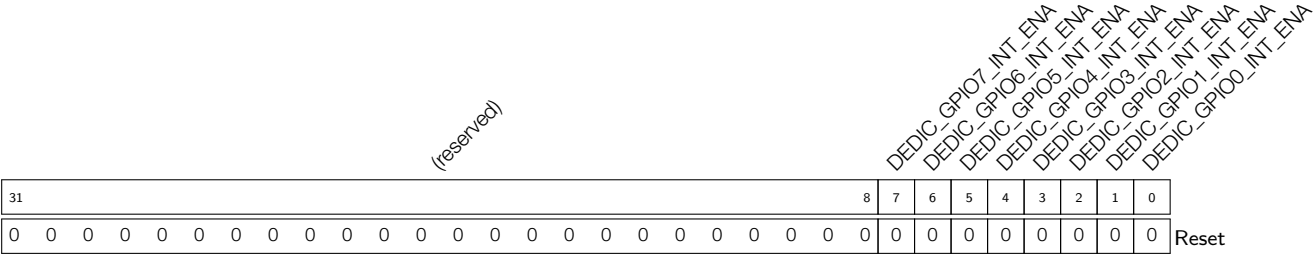
**DEDIC\_GPIO4\_INT\_RAW** This interrupt raw bit turns to high level when dedicated GPIO4 has level/edge change configured by [DEDIC\\_GPIO\\_INTR\\_RCGN\\_REG](#). (RO)

**DEDIC\_GPIO5\_INT\_RAW** This interrupt raw bit turns to high level when dedicated GPIO5 has level/edge change configured by [DEDIC\\_GPIO\\_INTR\\_RCGN\\_REG](#). (RO)

**DEDIC\_GPIO6\_INT\_RAW** This interrupt raw bit turns to high level when dedicated GPIO6 has level/edge change configured by [DEDIC\\_GPIO\\_INTR\\_RCGN\\_REG](#). (RO)

**DEDIC\_GPIO7\_INT\_RAW** This interrupt raw bit turns to high level when dedicated GPIO7 has level/edge change configured by [DEDIC\\_GPIO\\_INTR\\_RCGN\\_REG](#). (RO)

Register 5.49: DEDIC\_GPIO\_INTR\_RLS\_REG (0x0024)



- DEDIC\_GPIO0\_INT\_ENA** The enable bit for [DEDIC\\_GPIO0\\_INT\\_ST](#) register. (R/W)
- DEDIC\_GPIO1\_INT\_ENA** The enable bit for [DEDIC\\_GPIO1\\_INT\\_ST](#) register. (R/W)
- DEDIC\_GPIO2\_INT\_ENA** The enable bit for [DEDIC\\_GPIO2\\_INT\\_ST](#) register. (R/W)
- DEDIC\_GPIO3\_INT\_ENA** The enable bit for [DEDIC\\_GPIO3\\_INT\\_ST](#) register. (R/W)
- DEDIC\_GPIO4\_INT\_ENA** The enable bit for [DEDIC\\_GPIO4\\_INT\\_ST](#) register. (R/W)
- DEDIC\_GPIO5\_INT\_ENA** The enable bit for [DEDIC\\_GPIO5\\_INT\\_ST](#) register. (R/W)
- DEDIC\_GPIO6\_INT\_ENA** The enable bit for [DEDIC\\_GPIO6\\_INT\\_ST](#) register. (R/W)
- DEDIC\_GPIO7\_INT\_ENA** The enable bit for [DEDIC\\_GPIO7\\_INT\\_ST](#) register. (R/W)

**Register 5.50: DEDIC\_GPIO\_INTR\_ST\_REG (0x0028)**

(reserved)																								DEDIC_GPIO7_INT_ST DEDIC_GPIO6_INT_ST DEDIC_GPIO5_INT_ST DEDIC_GPIO4_INT_ST DEDIC_GPIO3_INT_ST DEDIC_GPIO2_INT_ST DEDIC_GPIO1_INT_ST DEDIC_GPIO0_INT_ST																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
31																								8	7	6	5	4	3	2	1	0	Reset																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**DEDIC\_GPIO0\_INT\_ST** This is the status bit for [DEDIC\\_GPIO0\\_INT\\_RAW](#) when [DEDIC\\_GPIO0\\_INT\\_ENA](#) is set to 1. (RO)

**DEDIC\_GPIO1\_INT\_ST** This is the status bit for [DEDIC\\_GPIO1\\_INT\\_RAW](#) when [DEDIC\\_GPIO1\\_INT\\_ENA](#) is set to 1. (RO)

**DEDIC\_GPIO2\_INT\_ST** This is the status bit for [DEDIC\\_GPIO2\\_INT\\_RAW](#) when [DEDIC\\_GPIO2\\_INT\\_ENA](#) is set to 1. (RO)

**DEDIC\_GPIO3\_INT\_ST** This is the status bit for [DEDIC\\_GPIO3\\_INT\\_RAW](#) when [DEDIC\\_GPIO3\\_INT\\_ENA](#) is set to 1. (RO)

**DEDIC\_GPIO4\_INT\_ST** This is the status bit for [DEDIC\\_GPIO4\\_INT\\_RAW](#) when [DEDIC\\_GPIO4\\_INT\\_ENA](#) is set to 1. (RO)

**DEDIC\_GPIO5\_INT\_ST** This is the status bit for [DEDIC\\_GPIO5\\_INT\\_RAW](#) when [DEDIC\\_GPIO5\\_INT\\_ENA](#) is set to 1. (RO)

**DEDIC\_GPIO6\_INT\_ST** This is the status bit for [DEDIC\\_GPIO6\\_INT\\_RAW](#) when [DEDIC\\_GPIO6\\_INT\\_ENA](#) is set to 1. (RO)

**DEDIC\_GPIO7\_INT\_ST** This is the status bit for [DEDIC\\_GPIO7\\_INT\\_RAW](#) when [DEDIC\\_GPIO7\\_INT\\_ENA](#) is set to 1. (RO)

Register 5.51: DEDIC\_GPIO\_INTR\_CLR\_REG (0x002C)

(reserved)																								DEDIC_GPIO7_INT_CLR DEDIC_GPIO6_INT_CLR DEDIC_GPIO5_INT_CLR DEDIC_GPIO4_INT_CLR DEDIC_GPIO3_INT_CLR DEDIC_GPIO2_INT_CLR DEDIC_GPIO1_INT_CLR DEDIC_GPIO0_INT_CLR									
31																								8	7	6	5	4	3	2	1	0	Reset
0 0																								0	0	0	0	0	0	0	0	0	

- DEDIC\_GPIO0\_INT\_CLR Set this bit to clear the [DEDIC\\_GPIO0\\_INT\\_RAW](#) interrupt. (WO)
- DEDIC\_GPIO1\_INT\_CLR Set this bit to clear the [DEDIC\\_GPIO1\\_INT\\_RAW](#) interrupt. (WO)
- DEDIC\_GPIO2\_INT\_CLR Set this bit to clear the [DEDIC\\_GPIO2\\_INT\\_RAW](#) interrupt. (WO)
- DEDIC\_GPIO3\_INT\_CLR Set this bit to clear the [DEDIC\\_GPIO3\\_INT\\_RAW](#) interrupt. (WO)
- DEDIC\_GPIO4\_INT\_CLR Set this bit to clear the [DEDIC\\_GPIO4\\_INT\\_RAW](#) interrupt. (WO)
- DEDIC\_GPIO5\_INT\_CLR Set this bit to clear the [DEDIC\\_GPIO5\\_INT\\_RAW](#) interrupt. (WO)
- DEDIC\_GPIO6\_INT\_CLR Set this bit to clear the [DEDIC\\_GPIO6\\_INT\\_RAW](#) interrupt. (WO)
- DEDIC\_GPIO7\_INT\_CLR Set this bit to clear the [DEDIC\\_GPIO7\\_INT\\_RAW](#) interrupt. (WO)

5.15.5 RTC IO MUX Registers

Register 5.52: RTCIO\_RTC\_GPIO\_OUT\_REG (0x0000)

RTCIO_GPIO_OUT_DATA										(reserved)													
31											10	9											0
0										0 0 0 0 0 0 0 0 0 0 0 0												Reset	

**RTCIO\_GPIO\_OUT\_DATA** GPIO0 ~ 21 output register. Bit10 corresponds to GPIO0, bit11 corresponds to GPIO1, etc. (R/W)



**Register 5.53: RTCIO\_RTC\_GPIO\_OUT\_W1TS\_REG (0x0004)**

RTCIO_GPIO_OUT_DATA_W1TS										(reserved)										
31										10	9									0
0										0 0 0 0 0 0 0 0 0 0										Reset

**RTCIO\_GPIO\_OUT\_DATA\_W1TS** GPIO0 ~ 21 output set register. If the value 1 is written to a bit here, the corresponding bit in [RTCIO\\_RTC\\_GPIO\\_OUT\\_REG](#) will be set to 1. Recommended operation: use this register to set [RTCIO\\_RTC\\_GPIO\\_OUT\\_REG](#). (WO)

**Register 5.54: RTCIO\_RTC\_GPIO\_OUT\_W1TC\_REG (0x0008)**

RTCIO_GPIO_OUT_DATA_W1TC										(reserved)										
31										10	9									0
0										0 0 0 0 0 0 0 0 0 0 0 0										Reset

**RTCIO\_GPIO\_OUT\_DATA\_W1TC** GPIO0 ~ 21 output clear register. If the value 1 is written to a bit here, the corresponding bit in [RTCIO\\_RTC\\_GPIO\\_OUT\\_REG](#) will be cleared. Recommended operation: use this register to clear [RTCIO\\_RTC\\_GPIO\\_OUT\\_REG](#). (WO)

**Register 5.55: RTCIO\_RTC\_GPIO\_ENABLE\_REG (0x000C)**

RTCIO_GPIO_ENABLE										(reserved)										
31										10	9									0
0										0 0 0 0 0 0 0 0 0 0 0 0										Reset

**RTCIO\_GPIO\_ENABLE** GPIO0 ~ 21 output enable. Bit10 corresponds to GPIO0, bit11 corresponds to GPIO1, etc. If the bit is set to 1, it means this GPIO pad is output. (R/W)

**Register 5.56: RTCIO\_RTC\_GPIO\_ENABLE\_W1TS\_REG (0x0010)**

RTCIO_GPIO_ENABLE_W1TS										(reserved)										
31										10	9									0
0										0 0 0 0 0 0 0 0 0 0 0 0										Reset

**RTCIO\_GPIO\_ENABLE\_W1TS** GPIO0 ~ 21 output enable set register. If the value 1 is written to a bit here, the corresponding bit in [RTCIO\\_RTC\\_GPIO\\_ENABLE\\_REG](#) will be set to 1. Recommended operation: use this register to set [RTCIO\\_RTC\\_GPIO\\_ENABLE\\_REG](#). (WO)

**Register 5.57: RTCIO\_RTC\_GPIO\_ENABLE\_W1TC\_REG (0x0014)**

RTCIO_GPIO_ENABLE_W1TC										(reserved)										
31										10	9									0
0										0 0 0 0 0 0 0 0 0 0 0 0										Reset

**RTCIO\_GPIO\_ENABLE\_W1TC** GPIO0 ~ 21 output enable clear register. If the value 1 is written to a bit here, the corresponding bit in [RTCIO\\_RTC\\_GPIO\\_ENABLE\\_REG](#) will be cleared. Recommended operation: use this register to clear [RTCIO\\_RTC\\_GPIO\\_ENABLE\\_REG](#). (WO)

**Register 5.58: RTCIO\_RTC\_GPIO\_STATUS\_REG (0x0018)**

RTCIO_GPIO_STATUS_INT										(reserved)										
31										10	9									0
0										0 0 0 0 0 0 0 0 0 0 0 0										Reset

**RTCIO\_GPIO\_STATUS\_INT** GPIO0 ~ 21 interrupt status register. Bit10 corresponds to GPIO0, bit11 corresponds to GPIO1, etc. This register should be used together with [RTCIO\\_RTC\\_GPIO\\_PIN<sub>n</sub>\\_INT\\_TYPE](#) in [RTCIO\\_RTC\\_GPIO\\_PIN<sub>n</sub>\\_REG](#). 0: no interrupt; 1: corresponding interrupt. (R/W)

**Register 5.59: RTCIO\_RTC\_GPIO\_STATUS\_W1TS\_REG (0x001C)**

RTCIO_GPIO_STATUS_INT_W1TS										(reserved)											
31										10	9										0
0										0 0 0 0 0 0 0 0 0 0										Reset	

**RTCIO\_GPIO\_STATUS\_INT\_W1TS** GPIO0 ~ 21 interrupt set register. If the value 1 is written to a bit here, the corresponding bit in [RTCIO\\_GPIO\\_STATUS\\_INT](#) will be set to 1. Recommended operation: use this register to set [RTCIO\\_GPIO\\_STATUS\\_INT](#). (WO)

**Register 5.60: RTCIO\_RTC\_GPIO\_STATUS\_W1TC\_REG (0x0020)**

RTCIO_GPIO_STATUS_INT_W1TC										(reserved)											
31										10	9										0
0										0 0 0 0 0 0 0 0 0 0										Reset	

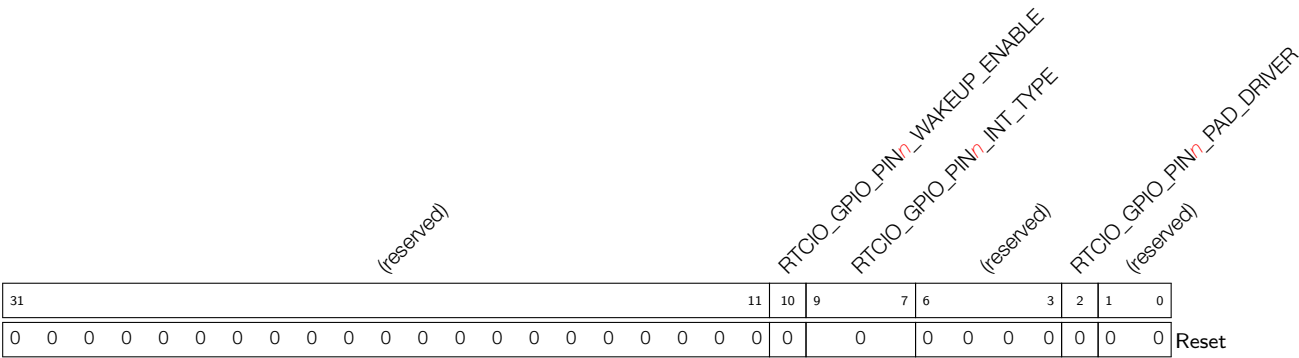
**RTCIO\_GPIO\_STATUS\_INT\_W1TC** GPIO0 ~ 21 interrupt clear register. If the value 1 is written to a bit here, the corresponding bit in [RTCIO\\_GPIO\\_STATUS\\_INT](#) will be cleared. Recommended operation: use this register to clear [RTCIO\\_GPIO\\_STATUS\\_INT](#). (WO)

**Register 5.61: RTCIO\_RTC\_GPIO\_IN\_REG (0x0024)**

RTCIO_GPIO_IN_NEXT										(reserved)											
31										10	9										0
0										0 0 0 0 0 0 0 0 0 0										Reset	

**RTCIO\_GPIO\_IN\_NEXT** GPIO0 ~ 21 input value. Bit10 corresponds to GPIO0, bit11 corresponds to GPIO1, etc. Each bit represents a pad input value, 1 for high level, and 0 for low level. (RO)

Register 5.62: RTCIO\_RTC\_GPIO\_PIN $n$ \_REG ( $n$ : 0-21) (0x0028+4\* $n$ )



- RTCIO\_GPIO\_PIN $n$ \_PAD\_DRIVER** Pad driver selection. 0: normal output; 1: open drain. (R/W)
- RTCIO\_GPIO\_PIN $n$ \_INT\_TYPE** GPIO interrupt type selection. 0: GPIO interrupt disabled; 1: rising edge trigger; 2: falling edge trigger; 3: any edge trigger; 4: low level trigger; 5: high level trigger. (R/W)
- RTCIO\_GPIO\_PIN $n$ \_WAKEUP\_ENABLE** GPIO wake-up enable. This will only wake up ESP32-S2 from Light-sleep. (R/W)

## 141

ESP32-S2 TRM (Preliminary V0.5)[Submit Documentation Feedback](#)

**RTCIO\_TOUCH\_PAD<sub>n</sub>\_DRV** Select the drive strength of the pad. 0: ~5 mA; 1: ~10 mA; 2: ~20 mA; 3: ~40 mA. (R/W)

## 142

ESP32-S2 TRM (Preliminary V0.5)

**RTCIO\_X32P\_DRV** Select the drive strength of the pad. 0: ~5 mA; 1: ~10 mA; 2: ~20 mA; 3: ~40 mA. (R/W)

## 143

ESP32-S2 TRM (Preliminary V0.5)

## 144

ESP32-S2 TRM (Preliminary V0.5)

**RTCIO\_PDAC1\_DRV** Select drive strength of the pad. 0: ~5 mA; 1: ~10 mA; 2: ~20 mA; 3: ~40 mA. (R/W)



## 145

ESP32-S2 TRM (Preliminary V0.5)

**RTCIO\_PDAC2\_DRV** Select drive strength of the pad. 0: ~5 mA; 1: ~10 mA; 2: ~20 mA; 3: ~40 mA. (R/W)

[Submit Documentation Feedback](#)

ESP32-S2 TRM (Preliminary V0.5)

**RTCIO\_RTC\_PAD19\_DRV** Select drive strength of the pad. 0: ~5 mA; 1: ~10 mA; 2: ~20 mA; 3: ~40 mA. (R/W)

[Submit Documentation Feedback](#)ESP32-S2 TRM (Preliminary V0.5)

**RTCIO\_RTC\_PAD20\_DRV** Select drive strength of the pad. 0: ~5 mA; 1: ~10 mA; 2: ~20 mA; 3: ~40 mA. (R/W)

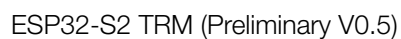
148

[Submit Documentation Feedback](#)

ESP32-S2 TRM (Preliminary V0.5)

**RTCIO\_XTL\_EXT\_CTR\_SEL** Select the external crystal power down enable source to get into sleep mode. 0: select GPIO0; 1: select GPIO1, etc. The input value on this pin XOR RTC\_CNTL\_EXT\_XTL\_CONF\_REG[30] is the crystal power down enable signal. (R/W)

## 149



## 6. System Registers

### 6.1 Overview

The ESP32-S2 integrates a large number of peripherals, and enables the control of individual peripherals to achieve optimal characteristics in performance-vs-power-consumption scenarios. Specifically, ESP32-S2 has a various of system configuration registers that can be used for the chip's clock management (clock gating), power management, and the configuration of peripherals and core-system modules. This chapter lists all these system registers and their functions.

### 6.2 Features

ESP32-S2 system registers can be used to control the following peripheral blocks and core modules:

- System and memory
- Reset and clock
- Interrupt matrix
- eFuse controller
- Low-power management
- Peripheral clock gating and reset

### 6.3 Function Description

#### 6.3.1 System and Memory Registers

The following registers are used for system and memory configuration, such as cache configuration and memory remapping. For additional information, please refer to Chapter 1 [System and Memory](#).

- [SYSTEM\\_ROM\\_CTRL\\_0\\_REG](#)
- [SYSTEM\\_ROM\\_CTRL\\_1\\_REG](#)
- [SYSTEM\\_SRAM\\_CTRL\\_0\\_REG](#)
- [SYSTEM\\_SRAM\\_CTRL\\_1\\_REG](#)
- [SYSTEM\\_SRAM\\_CTRL\\_2\\_REG](#)
- [SYSTEM\\_RSA\\_PD\\_CTRL\\_REG](#)
- [SYSTEM\\_MEM\\_PD\\_MASK\\_REG](#)
- [SYSTEM\\_CACHE\\_CONTROL\\_REG](#)
- [SYSTEM\\_BUSTOEXTMEM\\_ENA\\_REG](#)
- [SYSTEM\\_EXTERNAL\\_DEVICE\\_ENCRYPT\\_DECRYPT\\_CONTROL\\_REG](#)

## ROM Power Consumption Control

Registers [SYSTEM\\_ROM\\_CTRL\\_0\\_REG](#) and [SYSTEM\\_ROM\\_CTRL\\_1\\_REG](#) can be used to control the power consumption of ESP32-S2's ROM. Specifically:

- Setting different bits of the [SYSTEM\\_ROM\\_FO](#) field in register [SYSTEM\\_ROM\\_CTRL\\_0\\_REG](#) forces on the clock gates of different blocks of ROM.
- Setting different bits of the [SYSTEM\\_ROM\\_FORCE\\_PD](#) field in register [SYSTEM\\_ROM\\_CTRL\\_1\\_REG](#) powers down different blocks of internal ROM.
- Setting different bits of the [SYSTEM\\_ROM\\_FORCE\\_PU](#) field in register [SYSTEM\\_ROM\\_CTRL\\_1\\_REG](#) powers up different blocks of internal ROM.

For detailed information about the controlling bits of different blocks, please see Table 31 below.

**Table 31: ROM Controlling Bit**

ROM	Lowest Address1	Highest Address1	Lowest Address2	Highest Address2	Controlling Bit
Block0	0x4000_0000	0x4000_FFFF	-	-	Bit0
Block1	0x4001_2000	0x4001_FFFF	0x3FFA_0000	0x3FFA_FFFF	Bit1

## SRAM Power Consumption Control

Registers [SYSTEM\\_SRAM\\_CTRL\\_0\\_REG](#), [SYSTEM\\_SRAM\\_CTRL\\_1\\_REG](#), and [SYSTEM\\_SRAM\\_CTRL\\_2\\_REG](#) can be used to control the power consumption of ESP32-S2's internal SRAM. Specifically,

- Setting different bits of the [SYSTEM\\_SRAM\\_FO](#) field in register [SYSTEM\\_SRAM\\_CTRL\\_0\\_REG](#) forces on the clock gates of different blocks of internal SRAM.
- Setting different bits of the [SYSTEM\\_SRAM\\_FORCE\\_PD](#) field in register [SYSTEM\\_SRAM\\_CTRL\\_1\\_REG](#) powers down different blocks of internal SRAM.
- Setting different bits of the [SYSTEM\\_SRAM\\_FORCE\\_PU](#) field in register [SYSTEM\\_SRAM\\_CTRL\\_2\\_REG](#) powers up different blocks of internal SRAM.

For detailed information about the controlling bits of different blocks, please see Table 32 below.

**Table 32: SRAM Controlling Bit**

SRAM	Lowest Address1	Highest Address1	Lowest Address2	Highest Address2	Controlling Bit
Block0	0x4002_0000	0x4002_1FFF	0x3FFB_0000	0x3FFB_1FFF	Bit0
Block1	0x4002_2000	0x4002_3FFF	0x3FFB_2000	0x3FFB_3FFF	Bit1
Block2	0x4002_4000	0x4002_5FFF	0x3FFB_4000	0x3FFB_5FFF	Bit2
Block3	0x4002_6000	0x4002_7FFF	0x3FFB_6000	0x3FFB_7FFF	Bit3
Block4	0x4002_8000	0x4002_BFFF	0x3FFB_8000	0x3FFB_BFFF	Bit4
Block5	0x4002_C000	0x4002_FFFF	0x3FFB_C000	0x3FFB_FFFF	Bit5
Block6	0x4003_0000	0x4003_3FFF	0x3FFC_0000	0x3FFC_3FFF	Bit6
Block7	0x4003_4000	0x4003_7FFF	0x3FFC_4000	0x3FFC_7FFF	Bit7
Block8	0x4003_8000	0x4003_BFFF	0x3FFC_8000	0x3FFC_BFFF	Bit8
Block9	0x4003_C000	0x4003_FFFF	0x3FFC_C000	0x3FFC_FFFF	Bit9
Block10	0x4004_0000	0x4004_3FFF	0x3FFD_0000	0x3FFD_3FFF	Bit10

Block11	0x4004_4000	0x4004_7FFF	0x3FFD_4000	0x3FFD_7FFF	Bit11
Block12	0x4004_8000	0x4004_BFFF	0x3FFD_8000	0x3FFD_BFFF	Bit12
Block13	0x4004_C000	0x4004_FFFF	0x3FFD_C000	0x3FFD_FFFF	Bit13
Block14	0x4005_0000	0x4005_3FFF	0x3FFE_0000	0x3FFE_3FFF	Bit14
Block15	0x4005_4000	0x4005_7FFF	0x3FFE_4000	0x3FFE_7FFF	Bit15
Block16	0x4005_8000	0x4005_BFFF	0x3FFE_8000	0x3FFE_BFFF	Bit16
Block17	0x4005_C000	0x4005_FFFF	0x3FFE_C000	0x3FFE_FFFF	Bit17
Block18	0x4006_0000	0x4006_3FFF	0x3FFF_0000	0x3FFF_3FFF	Bit18
Block19	0x4006_4000	0x4006_7FFF	0x3FFF_4000	0x3FFF_7FFF	Bit19
Block20	0x4006_8000	0x4006_BFFF	0x3FFF_8000	0x3FFF_BFFF	Bit20
Block21	0x4006_C000	0x4006_FFFF	0x3FFF_C000	0x3FFF_FFFF	Bit21

### 6.3.2 Reset and Clock Registers

The following registers are used for reset and clock. For additional information, please refer to Chapter 2 [Reset and Clock](#).

- [SYSTEM\\_CPU\\_PER\\_CONF\\_REG](#)
- [SYSTEM\\_SYCLK\\_CONF\\_REG](#)
- [SYSTEM\\_BT\\_LPCK\\_DIV\\_FRAC\\_REG](#)

### 6.3.3 Interrupt Matrix Registers

The following registers are used for generating the CPU interrupt signals for the interrupt matrix. For additional information, please refer to Chapter 4 [Interrupt Matrix](#)

- [SYSTEM\\_CPU\\_INTR\\_FROM\\_CPU\\_0\\_REG](#)
- [SYSTEM\\_CPU\\_INTR\\_FROM\\_CPU\\_1\\_REG](#)
- [SYSTEM\\_CPU\\_INTR\\_FROM\\_CPU\\_2\\_REG](#)
- [SYSTEM\\_CPU\\_INTR\\_FROM\\_CPU\\_3\\_REG](#)

### 6.3.4 JTAG Software Enable Registers

The following registers are used for revoking the temporary disable of eFuse to JTAG.

- [SYSTEM\\_JTAG\\_CTRL\\_0\\_REG](#)
- [SYSTEM\\_JTAG\\_CTRL\\_1\\_REG](#)
- [SYSTEM\\_JTAG\\_CTRL\\_2\\_REG](#)
- [SYSTEM\\_JTAG\\_CTRL\\_3\\_REG](#)
- [SYSTEM\\_JTAG\\_CTRL\\_4\\_REG](#)
- [SYSTEM\\_JTAG\\_CTRL\\_5\\_REG](#)
- [SYSTEM\\_JTAG\\_CTRL\\_6\\_REG](#)
- [SYSTEM\\_JTAG\\_CTRL\\_7\\_REG](#)



### 6.3.5 Low-power Management Registers

The following registers are used for low-power management.

- [SYSTEM\\_RTC\\_FASTMEM\\_CONFIG\\_REG](#)
- [SYSTEM\\_RTC\\_FASTMEM\\_CRC\\_REG](#)

### 6.3.6 Peripheral Clock Gating and Reset Registers

The following registers are used for controlling the clock gating and reset of different peripherals. Details can be seen in Table 33.

- [SYSTEM\\_CPU\\_PERI\\_CLK\\_EN\\_REG](#)
- [SYSTEM\\_CPU\\_PERI\\_RST\\_EN\\_REG](#)
- [SYSTEM\\_PERIP\\_CLK\\_EN0\\_REG](#)
- [SYSTEM\\_PERIP\\_RST\\_EN0\\_REG](#)
- [SYSTEM\\_PERIP\\_CLK\\_EN1\\_REG](#)
- [SYSTEM\\_PERIP\\_RST\\_EN1\\_REG](#)

**Table 33: Peripheral Clock Gating and Reset Bits**

Peripheral	Clock Enabling Bit <sup>1</sup>	Reset Controlling Bit <sup>23</sup>
<b>CPU Peripherals</b>	<a href="#">SYSTEM_CPU_PERI_CLK_EN_REG</a>	<a href="#">SYSTEM_CPU_PERI_RST_EN_REG</a>
DEDICATED GPIO	<a href="#">SYSTEM_CLK_EN_DEDICATED_GPIO</a>	<a href="#">SYSTEM_RST_EN_DEDICATED_GPIO</a>
<b>Peripherals</b>	<a href="#">SYSTEM_PERIP_CLK_EN0_REG</a>	<a href="#">SYSTEM_PERIP_RST_EN0_REG</a>
Timers	<a href="#">SYSTEM_TIMERS_CLK_EN</a>	<a href="#">SYSTEM_TIMERS_RST</a>
Timer Group0	<a href="#">SYSTEM_TIMERGROUP_CLK_EN</a>	<a href="#">SYSTEM_TIMERGROUP_RST</a>
Timer Group1	<a href="#">SYSTEM_TIMERGROUP1_CLK_EN</a>	<a href="#">SYSTEM_TIMERGROUP1_RST</a>
System Timer	<a href="#">SYSTEM_SYSTIMER_CLK_EN</a>	<a href="#">SYSTEM_SYSTIMER_RST</a>
UART0	<a href="#">SYSTEM_UART_CLK_EN</a>	<a href="#">SYSTEM_UART_RST</a>
UART1	<a href="#">SYSTEM_UART1_CLK_EN</a>	<a href="#">SYSTEM_UART1_RST</a>
UART MEM	<a href="#">SYSTEM_UART_MEM_CLK_EN</a> <sup>4</sup>	<a href="#">SYSTEM_UART_MEM_RST</a>
SPI0, SPI1	<a href="#">SYSTEM_SPI01_CLK_EN</a>	<a href="#">SYSTEM_SPI01_RST</a>
SPI2	<a href="#">SYSTEM_SPI2_CLK_EN</a>	<a href="#">SYSTEM_SPI2_RST</a>
SPI3	<a href="#">SYSTEM_SPI3_DMA_CLK_EN</a>	<a href="#">SYSTEM_SPI3_RST</a>
SPI4	<a href="#">SYSTEM_SPI4_CLK_EN</a>	<a href="#">SYSTEM_SPI4_RST</a>
SPI2 DMA	<a href="#">SYSTEM_SPI2_DMA_CLK_EN</a>	<a href="#">SYSTEM_SPI2_DMA_RST</a>
SPI3 DMA	<a href="#">SYSTEM_SPI3_DMA_CLK_EN</a>	<a href="#">SYSTEM_SPI3_DMA_RST</a>
I2C0	<a href="#">SYSTEM_I2C_EXT0_CLK_EN</a>	<a href="#">SYSTEM_I2C_EXT0_RST</a>
I2C1	<a href="#">SYSTEM_I2C_EXT1_CLK_EN</a>	<a href="#">SYSTEM_I2C_EXT1_RST</a>
I2S0	<a href="#">SYSTEM_I2S0_CLK_EN</a>	<a href="#">SYSTEM_I2S0_RST</a>
I2S1	<a href="#">SYSTEM_I2S1_CLK_EN</a>	<a href="#">SYSTEM_I2S1_RST</a>
TWAI Controller	<a href="#">SYSTEM_CAN_CLK_EN</a>	<a href="#">SYSTEM_CAN_RST</a>
UHCI0	<a href="#">SYSTEM_UHCI0_CLK_EN</a>	<a href="#">SYSTEM_UHCI0_RST</a>
UHCI1	<a href="#">SYSTEM_UHCI1_CLK_EN</a>	<a href="#">SYSTEM_UHCI1_RST</a>
USB	<a href="#">SYSTEM_USB_CLK_EN</a>	<a href="#">SYSTEM_USB_RST</a>
RMT	<a href="#">SYSTEM_RMT_CLK_EN</a>	<a href="#">SYSTEM_RMT_RST</a>

PCNT	SYSTEM_PCNT_CLK_EN	SYSTEM_PCNT_RST
PWM0	SYSTEM_PWM0_CLK_EN	SYSTEM_PWM0_RST
PWM1	SYSTEM_PWM1_CLK_EN	SYSTEM_PWM1_RST
PWM2	SYSTEM_PWM2_CLK_EN	SYSTEM_PWM2_RST
PWM3	SYSTEM_PWM3_CLK_EN	SYSTEM_PWM3_RST
LED_PWM Controller	SYSTEM_LEDC_CLK_EN	SYSTEM_LEDC_RST
eFuse	SYSTEM_EFUSE_CLK_EN	SYSTEM_EFUSE_RST
APB SARADC	SYSTEM_APB_SARADC_CLK_EN	SYSTEM_APB_SARADC_RST
ADC2 ARB	SYSTEM_ADC2_ARB_CLK_EN	SYSTEM_ADC2_ARB_RST
WDG	SYSTEM_WDG_CLK_EN	SYSTEM_WDG_RST
<b>Accelerators</b>	<b>SYSTEM_PERIP_CLK_EN1_REG</b>	<b>SYSTEM_PERIP_RST_EN1_REG</b>
DMA	SYSTEM_CRYPTODMA_CLK_EN	SYSTEM_CRYPTODMA_RST <sup>5</sup>
HMAC	SYSTEM_CRYPTOHMAC_CLK_EN	SYSTEM_CRYPTOHMAC_RST <sup>6</sup>
Digital Signature	SYSTEM_CRYPTODS_CLK_EN	SYSTEM_CRYPTODS_RST <sup>7</sup>
RSA Accelerator	SYSTEM_CRYPTORSA_CLK_EN	SYSTEM_CRYPTORSA_RST
SHA Accelerator	SYSTEM_CRYPTOSHA_CLK_EN	SYSTEM_CRYPTOSHA_RST
AES Accelerator	SYSTEM_CRYPTOAES_CLK_EN	SYSTEM_CRYPTOAES_RST

**Note:**

1. Set the clock enable register to 1 to enable the clock, and to 0 to disable the clock;
2. Set the reset enabling register to 1 to reset a peripheral, and to 0 to disable the reset.
3. Reset registers are not cleared by hardware.
4. UART memory is shared by all UART peripherals, meaning having any active UART peripherals will prevent the UART memory from entering the clock-gated state.
5. Crypto DMA is shared by AES and SHA accelerators.
6. Resetting this bit also resets the SHA accelerator.
7. Resetting this bit also resets the AES, SHA, and RSA accelerators.

## 6.4 Base Address

Users can access the system registers with base address, which can be seen in the following table. For more information about accessing system registers, please see Chapter 1 *System and Memory*.

**Table 34: System Register Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F4C0000

## 6.5 Register Summary

The addresses in the following table are relative to the system registers base addresses provided in Section 6.4.

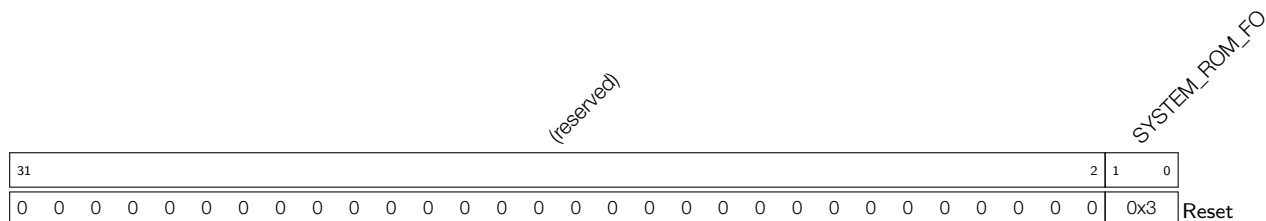
Name	Description	Address	Access
<b>System and Memory Registers</b>			
<a href="#">SYSTEM_ROM_CTRL_0_REG</a>	System ROM configuration register 0	0x0000	R/W
<a href="#">SYSTEM_ROM_CTRL_1_REG</a>	System ROM configuration register 1	0x0004	R/W
<a href="#">SYSTEM_SRAM_CTRL_0_REG</a>	System SRAM configuration register 0	0x0008	R/W
<a href="#">SYSTEM_SRAM_CTRL_1_REG</a>	System SRAM configuration register 1	0x000C	R/W
<a href="#">SYSTEM_SRAM_CTRL_2_REG</a>	System SRAM configuration register 2	0x0088	R/W
<a href="#">SYSTEM_MEM_PD_MASK_REG</a>	Memory power-related controlling register (under low-sleep)	0x003C	R/W
<a href="#">SYSTEM_RSA_PD_CTRL_REG</a>	RSA memory remapping register	0x0068	R/W
<a href="#">SYSTEM_BUSTOEXTMEM_ENA_REG</a>	EDMA enable register	0x006C	R/W
<a href="#">SYSTEM_CACHE_CONTROL_REG</a>	Cache control register	0x0070	R/W
<a href="#">SYSTEM_EXTERNAL_DEVICE_ENCRYPT_DECRYPT_CONTROL_REG</a>	External memory encrypt and decrypt controlling register	0x0074	R/W
<b>Reset and Clock Registers</b>			
<a href="#">SYSTEM_CPU_PER_CONF_REG</a>	CPU peripheral clock configuration register	0x0018	R/W
<a href="#">SYSTEM_BT_LPCK_DIV_FRAC_REG</a>	Divider fraction configuration register for low-power clock	0x0054	R/W
<a href="#">SYSTEM_SYSCLK_CONF_REG</a>	SoC clock configuration register	0x008C	Varies
<b>Interrupt Matrix Registers</b>			
<a href="#">SYSTEM_CPU_INTR_FROM_CPU_0_REG</a>	CPU interrupt controlling register 0	0x0058	R/W
<a href="#">SYSTEM_CPU_INTR_FROM_CPU_1_REG</a>	CPU interrupt controlling register 1	0x005C	R/W
<a href="#">SYSTEM_CPU_INTR_FROM_CPU_2_REG</a>	CPU interrupt controlling register 2	0x0060	R/W
<a href="#">SYSTEM_CPU_INTR_FROM_CPU_3_REG</a>	CPU interrupt controlling register 3	0x0064	R/W
<b>JTAG Software Enable Registers</b>			
<a href="#">SYSTEM_JTAG_CTRL_0_REG</a>	JTAG configuration register 0	0x001C	WO
<a href="#">SYSTEM_JTAG_CTRL_1_REG</a>	JTAG configuration register 1	0x0020	WO
<a href="#">SYSTEM_JTAG_CTRL_2_REG</a>	JTAG configuration register 2	0x0024	WO
<a href="#">SYSTEM_JTAG_CTRL_3_REG</a>	JTAG configuration register 3	0x0028	WO
<a href="#">SYSTEM_JTAG_CTRL_4_REG</a>	JTAG configuration register 4	0x002C	WO
<a href="#">SYSTEM_JTAG_CTRL_5_REG</a>	JTAG configuration register 5	0x0030	WO
<a href="#">SYSTEM_JTAG_CTRL_6_REG</a>	JTAG configuration register 6	0x0034	WO

Name	Description	Address	Access
<a href="#">SYSTEM_JTAG_CTRL_7_REG</a>	JTAG configuration register 7	0x0038	WO
<b>Low-Power Management Registers</b>			
<a href="#">SYSTEM_RTC_FASTMEM_CONFIG_REG</a>	RTC fast memory configuration register	0x0078	Varies
<a href="#">SYSTEM_RTC_FASTMEM_CRC_REG</a>	RTC fast memory CRC controlling register	0x007C	RO
<b>Peripheral Clock Gating and Reset Registers</b>			
<a href="#">SYSTEM_CPU_PERI_CLK_EN_REG</a>	CPU peripheral clock enable register	0x0010	R/W
<a href="#">SYSTEM_CPU_PERI_RST_EN_REG</a>	CPU peripheral reset register	0x0014	R/W
<a href="#">SYSTEM_PERIP_CLK_EN0_REG</a>	System peripheral clock (for hardware accelerators) enable register 0	0x0040	R/W
<a href="#">SYSTEM_PERIP_CLK_EN1_REG</a>	System peripheral clock (for hardware accelerators) enable register 1	0x0044	R/W
<a href="#">SYSTEM_PERIP_RST_EN0_REG</a>	System peripheral (hardware accelerators) reset register 0	0x0048	R/W
<a href="#">SYSTEM_PERIP_RST_EN1_REG</a>	System peripheral (hardware accelerators) reset register 1	0x004C	R/W
<b>Version Register</b>			
<a href="#">SYSTEM_DATE_REG</a>	Version control register	0x0FFC	R/W

## 6.6 Registers

The addresses below are relative to the system registers base addresses provided in Section 6.4.

**Register 6.1: SYSTEM\_ROM\_CTRL\_0\_REG (0x0000)**



**SYSTEM\_ROM\_FO** This field is used to force on clock gate of internal ROM. For details, please refer to Table 31. (R/W)

Register 6.2: SYSTEM\_ROM\_CTRL\_1\_REG (0x0004)

(reserved)																								SYSTEM_ROM_FORCE_PU				SYSTEM_ROM_FORCE_PD			
31																								4	3	2	1	0			
0 0																								3	0	Reset					

**SYSTEM\_ROM\_FORCE\_PD** This field is used to power down internal ROM. For details, please refer to Table 31. (R/W)

**SYSTEM\_ROM\_FORCE\_PU** This field is used to power up internal ROM. For details, please refer to Table 31. (R/W)

Register 6.3: SYSTEM\_SRAM\_CTRL\_0\_REG (0x0008)

(reserved)											SYSTEM_SRAM_FO										
3122											210										
0000000000											0x3ffff										Reset

**SYSTEM\_SRAM\_FO** This field is used to force on clock gate of internal SRAM. For details, please refer to Table 32. (R/W)

Register 6.4: SYSTEM\_SRAM\_CTRL\_1\_REG (0x000C)

(reserved)											SYSTEM_SRAM_FORCE_PD										
31											22	21									0
0	0	0	0	0	0	0	0	0	0	0	0										Reset

**SYSTEM\_SRAM\_FORCE\_PD** This field is used to power down internal SRAM. For details, please refer to Table 32. (R/W)

Register 6.5: SYSTEM\_CPU\_PERI\_CLK\_EN\_REG (0x0010)

(reserved)																SYSTEM_CLK_EN_DEDICATED_GPIO				(reserved)					
31																	8	7	6	5					0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

**SYSTEM\_CLK\_EN\_DEDICATED\_GPIO** Set this bit to enable clock of DEDICATED GPIO module.

Register 6.6: SYSTEM\_CPU\_PERI\_RST\_EN\_REG (0x0014)

(reserved)																								SYSTEM_RST_EN_DEDICATED_GPIO				(reserved)			
31																							8	7	6	5					0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	Reset

**SYSTEM\_RST\_EN\_DEDICATED\_GPIO** Set this bit to reset DEDICATED GPIO module. (R/W)

Register 6.7: SYSTEM\_CPU\_PER\_CONF\_REG (0x0018)

(reserved)																								SYSTEM_CPU_WAITI_DELAY_NUM				SYSTEM_CPU_WAIT_MODE_FORCE_ON		SYSTEM_PLL_FREQ_SEL		SYSTEM_CPUPERIOD_SEL					
31																								8				7	4				3	2	1	0	
0 0																								0x0				1	1	0	Reset						

**SYSTEM\_CPUPERIOD\_SEL** This field is used to select the clock frequency of CPU or CPU period.

For details, please refer to Table 8 in Chapter 2 *Reset and Clock*. (R/W)

**SYSTEM\_PLL\_FREQ\_SEL** This field is used to select the PLL clock frequency based on CPU period.

For details, please refer to Table 8 in Chapter 2 *Reset and Clock*. (R/W)

**SYSTEM\_CPU\_WAIT\_MODE\_FORCE\_ON** Set this bit to force on CPU wait mode. In this mode, the clock gate of CPU is turned off until any interrupts happen. This mode could also be force on via WAITI instruction. (R/W)

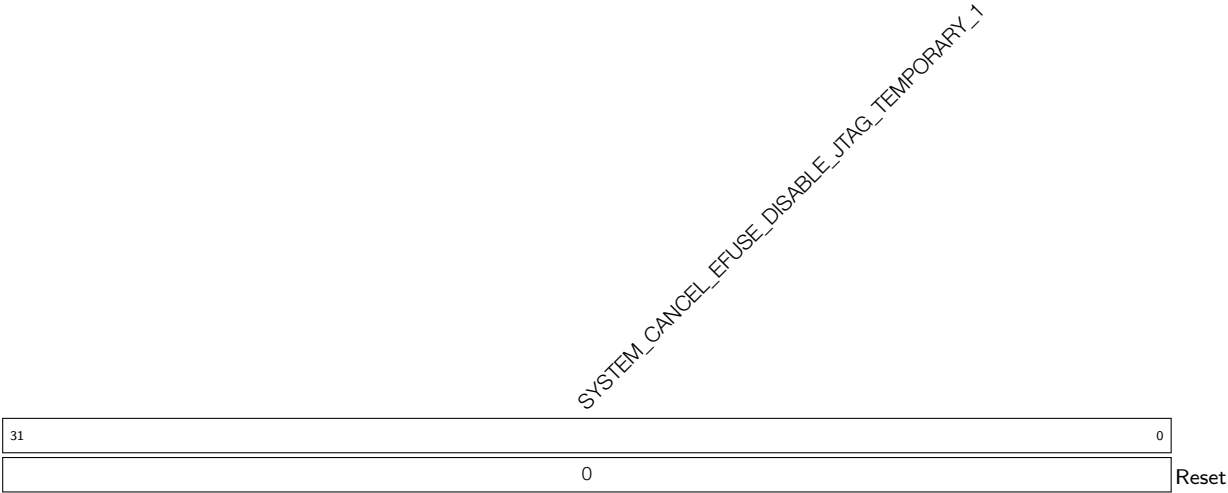
**SYSTEM\_CPU\_WAITI\_DELAY\_NUM** Sets the number of delay cycles to enter CPU wait mode after a WAITI instruction. (R/W)

Register 6.8: SYSTEM\_JTAG\_CTRL\_0\_REG (0x001C)

SYSTEM_CANCEL_EFUSE_DISABLE_JTAG_TEMPORARY_0																															
31																															0
0																															
Reset																															

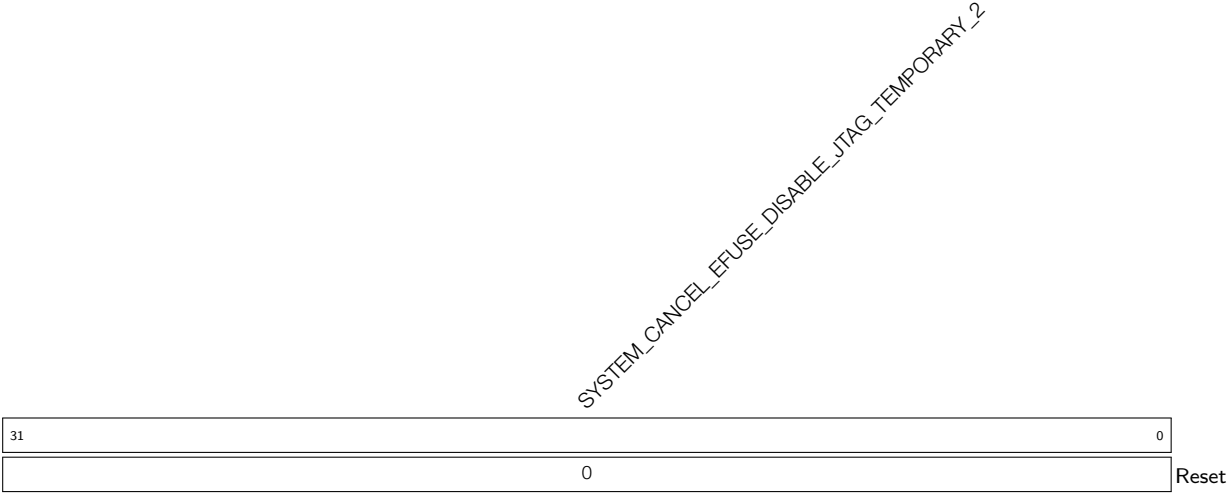
**SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_0** Stores the 0 to 31 bits of the 256 bits register used to cancel the temporary disable of eFuse to JTAG.

Register 6.9: SYSTEM\_JTAG\_CTRL\_1\_REG (0x0020)



**SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_1** Stores the 32 to 63 bits of the 256 bits register used to cancel the temporary disable of eFuse to JTAG.

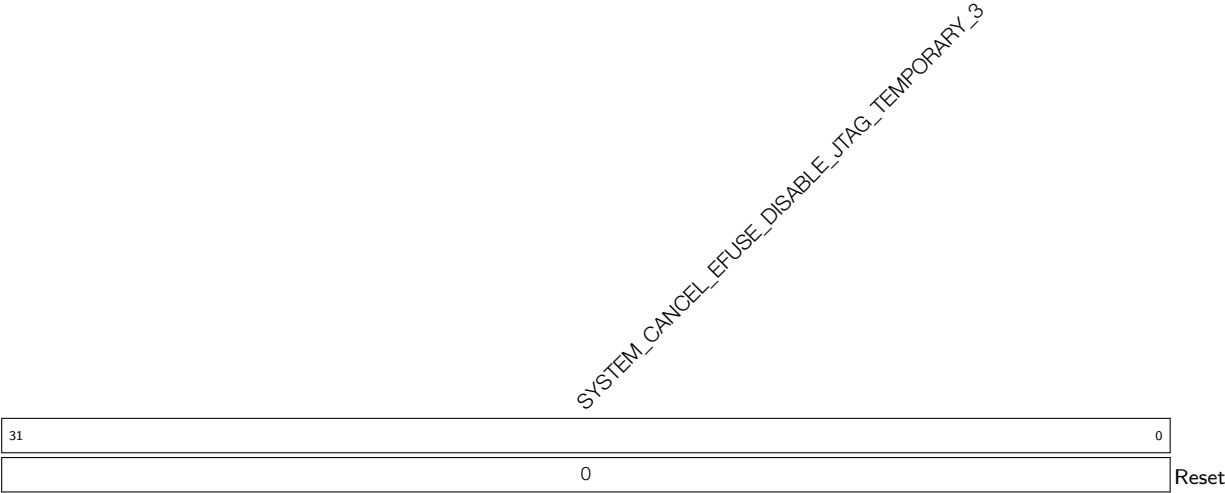
Register 6.10: SYSTEM\_JTAG\_CTRL\_2\_REG (0x0024)



**SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_2** Stores the 64 to 95 bits of the 256 bits register used to cancel the temporary disable of eFuse to JTAG.

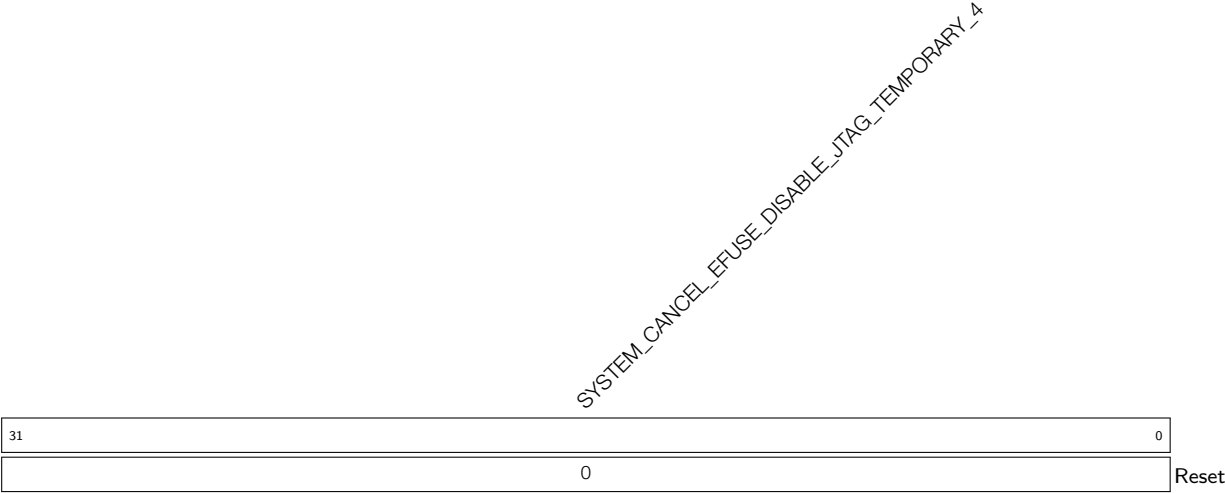


Register 6.11: SYSTEM\_JTAG\_CTRL\_3\_REG (0x0028)



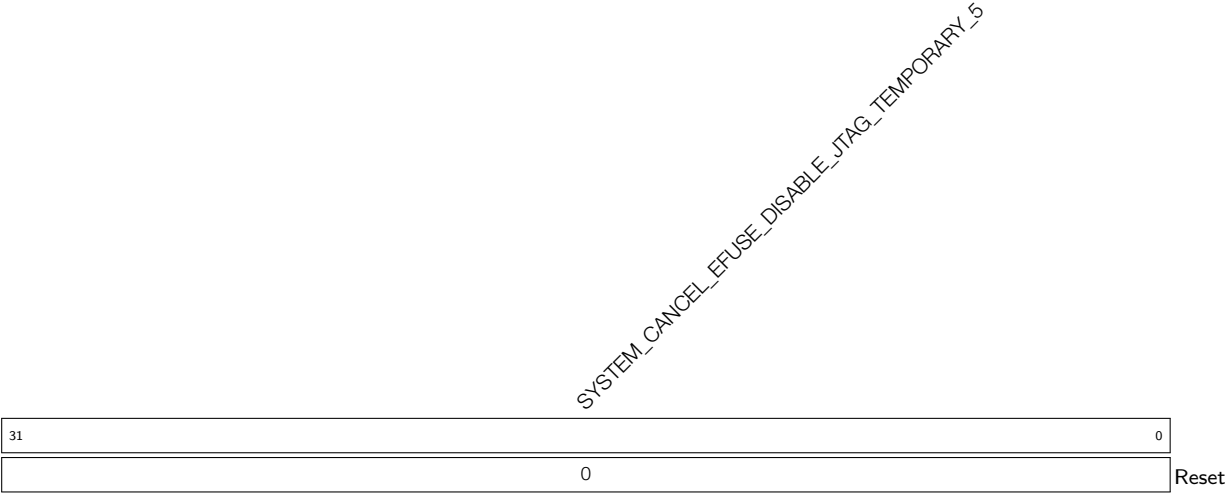
**SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_3** Stores the 96 to 127 bits of the 256 bits register used to cancel the temporary disable of eFuse to JTAG.

Register 6.12: SYSTEM\_JTAG\_CTRL\_4\_REG (0x002C)



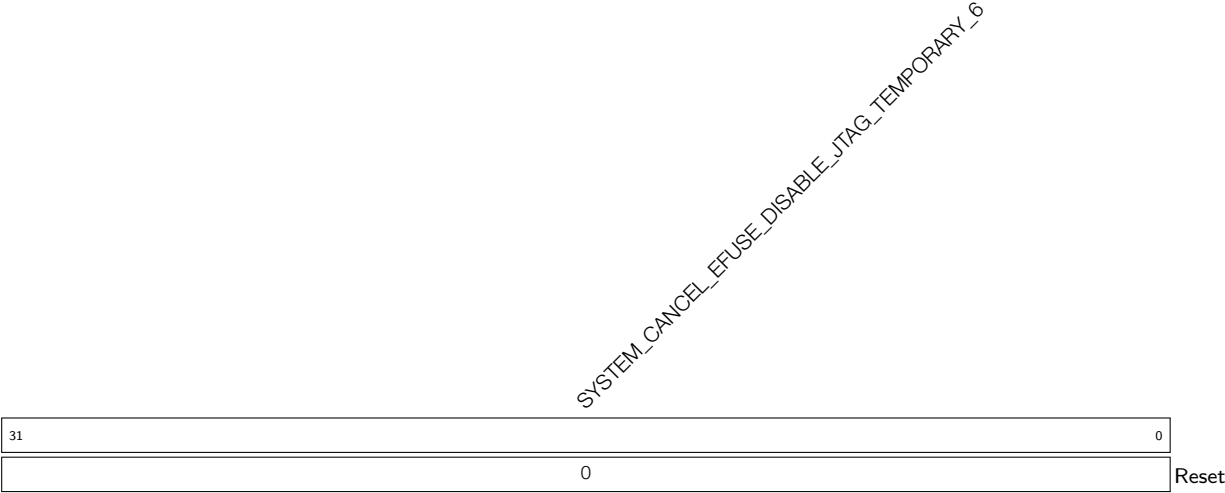
**SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_4** Stores the 128 to 159 bits of the 256 bits register used to cancel the temporary disable of eFuse to JTAG.

Register 6.13: SYSTEM\_JTAG\_CTRL\_5\_REG (0x0030)



**SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_5** Stores the 160 to 191 bits of the 256 bits register used to cancel the temporary disable of eFuse to JTAG.

Register 6.14: SYSTEM\_JTAG\_CTRL\_6\_REG (0x0034)



**SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_6** Stores the 192 to 223 bits of the 256 bits register used to cancel the temporary disable of eFuse to JTAG.

### Register 6.15: SYSTEM\_JTAG\_CTRL\_7\_REG (0x0038)

**Register 6.18: SYSTEM\_PERIP\_CLK\_EN1\_REG (0x0044)**

(reserved)																								SYSTEM_CRYPT0_DMA_CLK_EN SYSTEM_CRYPT0_HMAC_CLK_EN SYSTEM_CRYPT0_DS_CLK_EN SYSTEM_CRYPT0_RSA_CLK_EN SYSTEM_CRYPT0_SHA_CLK_EN SYSTEM_CRYPT0_AES_CLK_EN (reserved)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
31																								7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**SYSTEM\_PERIP\_CLK\_EN1\_REG** Configures this register to enable different accelerator clocks. For details, please refer to Table 33.

**Register 6.19: SYSTEM\_PERIP\_RST\_EN0\_REG (0x0048)**

SYSTEM_SPI4_RST SYSTEM_ADC2_ARB_RST SYSTEM_SYSTIMER_RST SYSTEM_APB_SARADC_RST SYSTEM_SPI3_DMA_RST SYSTEM_PWM3_RST SYSTEM_PWM2_RST SYSTEM_UART_MEM_RST SYSTEM_USB_RST SYSTEM_SPI2_DMA_RST SYSTEM_CAN_RST SYSTEM_I2C_EXT1_RST SYSTEM_PWM0_RST SYSTEM_SPI3_RST SYSTEM_TIMERGROUP1_RST SYSTEM_TIMERGROUP_RST SYSTEM_UHCI1_RST SYSTEM_LEDC_RST SYSTEM_PONT_RST SYSTEM_RMT_RST SYSTEM_UHCI0_RST SYSTEM_I2C_EXT0_RST SYSTEM_SPI2_RST SYSTEM_UART1_RST SYSTEM_I2S0_RST SYSTEM_WDG_RST SYSTEM_UART_RST SYSTEM_SPI01_RST SYSTEM_TIMERS_RST																																
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	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**SYSTEM\_PERIP\_RST\_EN0\_REG** Configures this register to reset different peripherals. For details, please refer to Table 33.

**Register 6.20: SYSTEM\_PERIP\_RST\_EN1\_REG (0x004C)**

(reserved)																								SYSTEM_CRYPT0_DMA_RST SYSTEM_CRYPT0_HMAC_RST SYSTEM_CRYPT0_DS_RST SYSTEM_CRYPT0_RSA_RST SYSTEM_CRYPT0_SHA_RST SYSTEM_CRYPT0_AES_RST (reserved)							
31																								7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	Reset

**SYSTEM\_PERIP\_RST\_EN1\_REG** Configures this register to reset different accelerators. For details, please refer to Table 33.

Register 6.21: SYSTEM\_BT\_LPCK\_DIV\_FRAC\_REG (0x0054)

(reserved)								SYSTEM_LPCLK_RTC_EN																SYSTEM_LPCLK_SEL_XTAL32K																SYSTEM_LPCLK_SEL_XTAL																SYSTEM_LPCLK_SEL_8M																(reserved)								(reserved)															
31		29		28		27		26		25		24		23																		12		11																0																																													
0		0		0		0		0		0		1		0		1																1																Reset																																															

Reset

**SYSTEM\_LPCLK\_SEL\_RTC\_SLOW** Set this bit to select RTC slow clock as the low power clock.

(R/W)

**SYSTEM\_LPCLK\_SEL\_8M** Set this bit to select 8m clock as the low power clock. (R/W)

**SYSTEM\_LPCLK\_SEL\_XTAL** Set this bit to select xtal clock as the low power clock. (R/W)

**SYSTEM\_LPCLK\_SEL\_XTAL32K** Set this bit to select xtal32k clock as the low power clock. (R/W)

**SYSTEM\_LPCLK\_RTC\_EN** Set this bit to enable the RTC low power clock. (R/W)

Register 6.22: SYSTEM\_CPU\_INTR\_FROM\_CPU\_0\_REG (0x0058)

(reserved)																												SYSTEM_CPU_INTR_FROM_CPU_0	
31																												1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

**SYSTEM\_CPU\_INTR\_FROM\_CPU\_0** Set this bit to generate CPU interrupt 0. This bit needs to be reset by software in the ISR process. (R/W)

Register 6.23: SYSTEM\_CPU\_INTR\_FROM\_CPU\_1\_REG (0x005C)

(reserved)																																SYSTEM_CPU_INTERRUPT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

Reset

**SYSTEM\_CPU\_INTR\_FROM\_CPU\_1** Set this bit to generate CPU interrupt 1. This bit needs to be reset by software in the ISR process. (R/W)

Register 6.24: SYSTEM\_CPU\_INTR\_FROM\_CPU\_2\_REG (0x0060)

(reserved)																															SYSTEM_CPU_INTR_FROM_CPU_2																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
31																														1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**SYSTEM\_CPU\_INTR\_FROM\_CPU\_2** Set this bit to generate CPU interrupt 2. This bit needs to be reset by software in the ISR process. (R/W)

Register 6.25: SYSTEM\_CPU\_INTR\_FROM\_CPU\_3\_REG (0x0064)

(reserved)																															SYSTEM_CPU_INTR_FROM_CPU_3																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
31																														1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

**SYSTEM\_CPU\_INTR\_FROM\_CPU\_3** Set this bit to generate CPU interrupt 3. This bit needs to be reset by software in the ISR process. (R/W)

Register 6.26: SYSTEM\_RSA\_PD\_CTRL\_REG (0x0068)

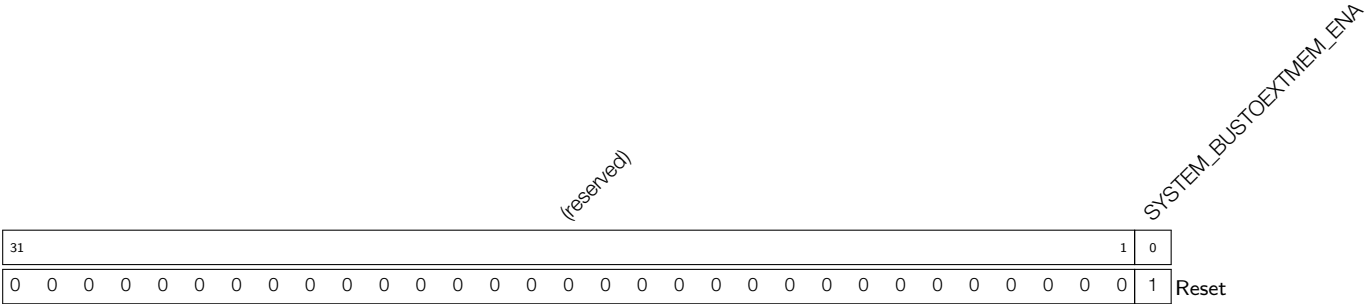
(reserved)																															SYSTEM_RSA_MEM_FORCE_PD SYSTEM_RSA_MEM_FORCE_PU SYSTEM_RSA_MEM_PD					
31																															3	2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Reset

**SYSTEM\_RSA\_MEM\_PD** Set this bit to power down RSA memory. This bit has the lowest priority. When Digital Signature occupies the RSA, this bit is invalid. (R/W)

**SYSTEM\_RSA\_MEM\_FORCE\_PU** Set this bit to force power up RSA memory. This bit has the second highest priority. (R/W)

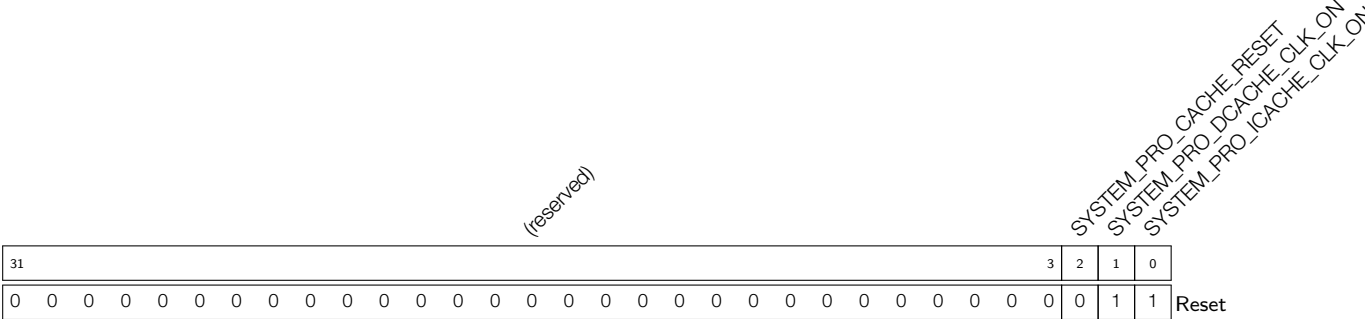
**SYSTEM\_RSA\_MEM\_FORCE\_PD** Set this bit to force power down RSA memory. This bit has the highest priority. (R/W)

Register 6.27: SYSTEM\_BUSTOEXTMEM\_ENA\_REG (0x006C)



**SYSTEM\_BUSTOEXTMEM\_ENA** Set this bit to enable bus to EDMA. (R/W)

Register 6.28: SYSTEM\_CACHE\_CONTROL\_REG (0x0070)



**SYSTEM\_PRO\_ICACHE\_CLK\_ON** Set this bit to enable clock of i-cache. (R/W)

**SYSTEM\_PRO\_DCACHE\_CLK\_ON** Set this bit to enable clock of d-cache. (R/W)

**SYSTEM\_PRO\_CACHE\_RESET** Set this bit to reset cache. (R/W)

**Register 6.29: SYSTEM\_EXTERNAL\_DEVICE\_ENCRYPT\_DECRYPT\_CONTROL\_REG (0x0074)**

(reserved)																												SYSTEM_ENABLE_DOWNLOAD_MANUAL_ENCRYPT SYSTEM_ENABLE_DOWNLOAD_G0CB_DECRYPT SYSTEM_ENABLE_DOWNLOAD_DB_ENCRYPT SYSTEM_ENABLE_SPI_MANUAL_ENCRYPT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
31																												4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**SYSTEM\_ENABLE\_SPI\_MANUAL\_ENCRYPT** Set this bit to enable Manual Encryption under SPI Boot mode. (R/W)

**SYSTEM\_ENABLE\_DOWNLOAD\_DB\_ENCRYPT** Set this bit to enable Auto Encryption under Download Boot mode. (R/W)

**SYSTEM\_ENABLE\_DOWNLOAD\_G0CB\_DECRYPT** Set this bit to enable Auto Decryption under Download Boot mode. (R/W)

**SYSTEM\_ENABLE\_DOWNLOAD\_MANUAL\_ENCRYPT** Set this bit to enable Manual Encryption under Download Boot mode. (R/W)

**Register 6.30: SYSTEM\_RTC\_FASTMEM\_CONFIG\_REG (0x0078)**

SYSTEM_RTC_MEM_CRC_FINISH										SYSTEM_RTC_MEM_CRC_LEN										SYSTEM_RTC_MEM_CRC_ADDR										SYSTEM_RTC_MEM_CRC_START										(reserved)																					
31	30																			20	19	9																			8	7	0																		
0	0x7ff																			0x0																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset								

**SYSTEM\_RTC\_MEM\_CRC\_START** Set this bit to start the CRC of RTC memory. (R/W)

**SYSTEM\_RTC\_MEM\_CRC\_ADDR** This field is used to set address of RTC memory for CRC. (R/W)

**SYSTEM\_RTC\_MEM\_CRC\_LEN** This field is used to set length of RTC memory for CRC based on start address. (R/W)

**SYSTEM\_RTC\_MEM\_CRC\_FINISH** This bit stores the status of RTC memory CRC. High level means finished while low level means not finished. (RO)



**Register 6.31: SYSTEM\_RTC\_FASTMEM\_CRC\_REG (0x007C)**

SYSTEM_RTC_MEM_CRC_RES																															
31																															0
0																															
																															Reset

**SYSTEM\_RTC\_MEM\_CRC\_RES** This field stores the CRC result of RTC memory. (RO)

**Register 6.32: SYSTEM\_SRAM\_CTRL\_2\_REG (0x0088)**

(reserved)											SYSTEM_SRAM_FORCE_PU																															
31											22											21																				0
0 0 0 0 0 0 0 0 0 0											0x3ffff																				Reset											

**SYSTEM\_SRAM\_FORCE\_PU** This field is used to power up internal SRAM. For details, please refer to Table 32. (R/W)

**Register 6.33: SYSTEM\_SYSCLK\_CONF\_REG (0x008C)**

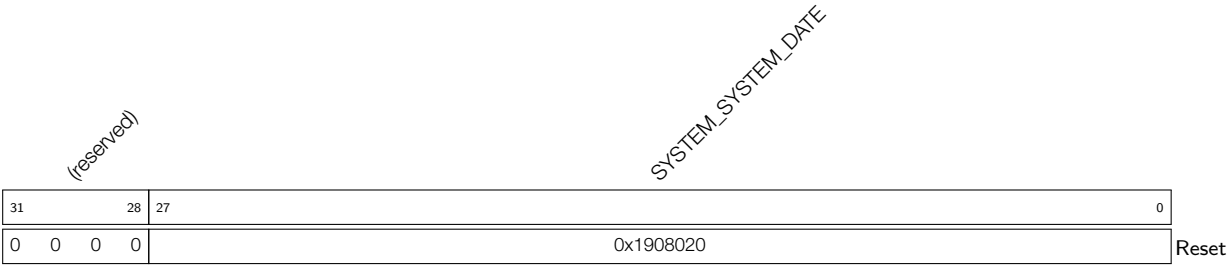
(reserved)												SYSTEM_CLK_XTAL_FREQ					SYSTEM_SOC_CLK_SEL			SYSTEM_PRE_DIV_CNT										
31												19	18				12	11	10	9										0
0	0	0	0	0	0	0	0	0	0	0	0	0			0			0x1											Reset	

**SYSTEM\_PRE\_DIV\_CNT** This field is used to set the count of prescaler of XTAL\_CLK. For details, please refer to Table 10 in Chapter 2 *Reset and Clock*. (R/W)

**SYSTEM\_SOC\_CLK\_SEL** This field is used to select SOC clock. For details, please refer to Table 8 in Chapter 2 *Reset and Clock*. (R/W)

**SYSTEM\_CLK\_XTAL\_FREQ** This field is used to read XTAL frequency in MHz. (RO)

Register 6.34: SYSTEM\_DATE\_REG (0x0FFC)



**SYSTEM\_DATE** Version control register. (R/W)

## 7. DMA Controller

### 7.1 Overview

Direct Memory Access (DMA) is a feature that allows peripheral-to-memory and memory-to-memory data transfer at a high speed. The CPU is not involved in the DMA transfer, and therefore it becomes more efficient.

ESP32-S2 has three types of DMA, namely Internal DMA, EDMA and Copy DMA. Internal DMA can only access internal RAM and is used for data transfer between internal RAM and peripherals. EDMA can access both internal RAM and external RAM and is used for data transfer between internal RAM, external RAM and peripherals. Copy DMA can only access internal RAM and is used for data transfer from one location in internal RAM to another.

Eight peripherals on ESP32-S2 have DMA features. As shown in Figure 7-1, UART0 and UART1 share one Internal DMA; SPI3 and ADC Controller share one Internal DMA; AES Accelerator and SHA Accelerator share one EDMA; SPI2 and I2S0 have their individual EDMA. Besides, the CPU Peripheral module on ESP32-S2 also has one Copy DMA.

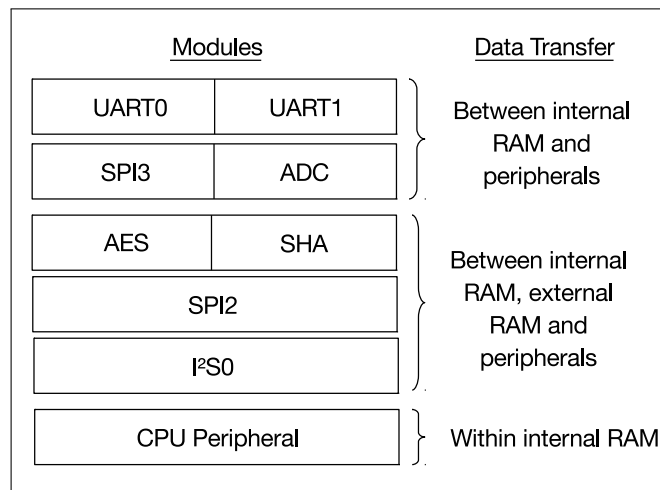


Figure 7-1. Modules with DMA and Supported Data Transfers

### 7.2 Features

The DMA controller has the following features:

- AHB bus architecture
- Half-duplex and full-duplex mode
- Programmable length of data to be transferred in bytes
- INCR burst transfer when accessing internal RAM
- Access to an address space of 320 KB at most in internal RAM
- Access to an address space of 10.5 MB at most in external RAM
- High-speed data transfer using DMA

## 7.3 Functional Description

In ESP32-S2, all modules that need high-speed data transfer support DMA. The DMA controller and CPU data bus have access to the same address space in internal RAM and external RAM. DMA controllers for different modules vary in functions according to needs, but their architecture is identical.

### 7.3.1 DMA Engine Architecture

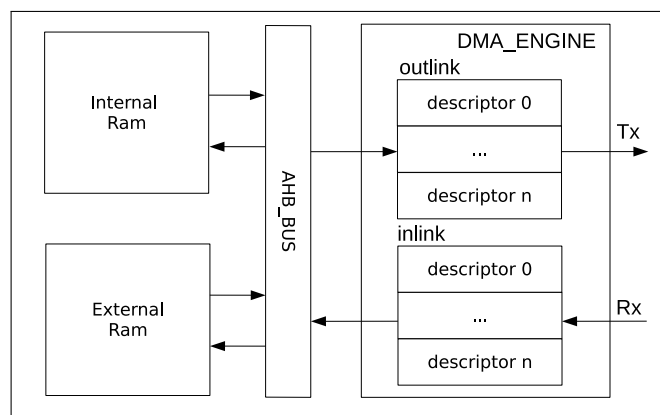


Figure 7-2. DMA Engine Architecture

A DMA engine reads/writes data to/from external RAM or internal RAM via the AHB\_BUS. Figure 7-2 shows the basic architecture of a DMA engine. For how to access RAM, please see Chapter 1 *System and Memory*. Software can use the DMA engine through linked lists. The DMA\_ENGINE transmits data in corresponding RAM according to the outlink (i.e. a linked list of transmit descriptors), and stores received data into specific address space in RAM according to the inlink (i.e. a linked list of receive descriptors).

### 7.3.2 Linked List

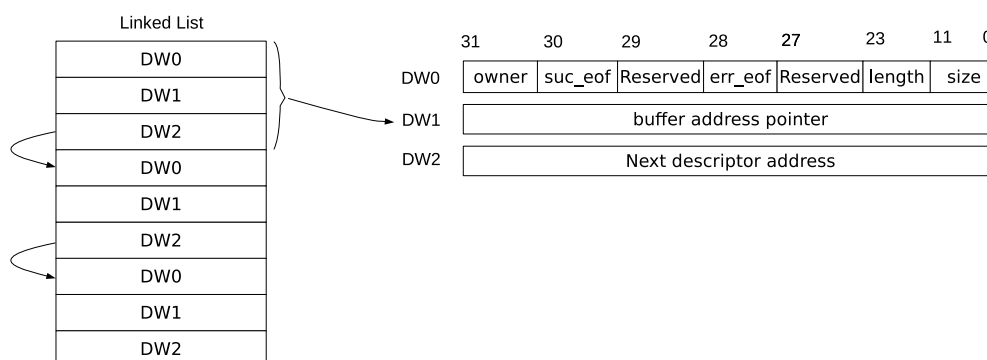


Figure 7-3. Structure of a Linked List

Figure 7-3 shows the structure of a linked list. An outlink and an inlink have the same structure. A linked list is formed by one or more descriptors, and each descriptor consists of three words. Linked lists should be in internal RAM for the DMA engine to be able to use them. The meaning of each field is as follows:

- Owner (DW0) [31]: Specifies who is allowed to access the buffer that this descriptor points to.  
1'b0: CPU can access the buffer;  
1'b1: The DMA controller can access the buffer.

When the DMA controller stops using the buffer, this bit is cleared by hardware. You can set

*PERI\_IN\_LOOP\_TEST* bit to disable automatic clearing by hardware. When software loads a linked list, this

bit should be set to 1.

**Note:** *PERI* refers to modules that support DMA transfers, e.g. I2S, SPI, UHCI, etc.

- `suc_eof` (DW0) [30]: Specifies whether this descriptor is the last descriptor in the list.
  - 1'b0: This descriptor is not the last one;
  - 1'b1: This descriptor is the last one.Software clears `suc_eof` bit in receive descriptors. When a packet has been received, this bit in the last receive descriptor is set by hardware, and this bit in the last transmit descriptor is set by software.
- Reserved (DW0) [29]: Reserved.
- `err_eof` (DW0) [28]: Specifies whether the received data has errors.
  - This bit is used only when UART DMA receives data. When an error is detected in the received packet, this bit in the receive descriptor is set to 1 by hardware.
- Reserved (DW0) [27:24]: Reserved.
- Length (DW0) [23:12]: Specifies the number of valid bytes in the buffer that this descriptor points to. This field in a transmit descriptor is written by software and indicates how many bytes can be read from the buffer; this field in a receive descriptor is written by hardware automatically and indicates how many bytes have been stored into the buffer.
- Size (DW0) [11:0]: Specifies the size of the buffer that this descriptor points to.
  - When the DMA controller accesses external RAM, this field must be a multiple of 16/32/64 bytes. Please see more details in Section [7.3.8 Accessing External RAM](#).
- Buffer address pointer (DW1): Pointer to the buffer.
  - When the DMA controller accesses external RAM, the destination address must be aligned with *PERI\_EXT\_MEM\_BK\_SIZE* field. Please see more details in Section [7.3.8 Accessing External RAM](#).
- Next descriptor address (DW2): Pointer to the next descriptor. If the current descriptor is the last one (`suc_eof` = 1), this value is 0. This field can only point to internal RAM.

If the length of data received is smaller than the size of the buffer, the DMA controller will not use available space of the buffer in the next transaction.

### 7.3.3 Enabling DMA

Software uses the DMA controller through linked lists. When the DMA controller receives data, software loads an inlink, configures *PERI\_INLINK\_ADDR* field with address of the first receive descriptor, and sets *PERI\_INLINK\_START* bit to enable DMA. When the DMA controller transmits data, software loads an outlink, prepares data to be transmitted, configures *PERI\_OUTLINK\_ADDR* field with address of the first transmit descriptor, and sets *PERI\_OUTLINK\_START* bit to enable DMA. *PERI\_INLINK\_START* bit and *PERI\_OUTLINK\_START* bit are cleared automatically by hardware.

The DMA controller can be restarted. If you are not sure whether the loaded linked list has been used up or not and want to load a new linked list, you can use this Restart function without affecting the loaded linked list. When using the Restart function, software needs to rewrite address of the first descriptor in the new list to DW2 of the last descriptor in the loaded list, loads the new list as shown in Figure [7-4](#), and set *PERI\_INLINK\_RESTART* bit or *PERI\_OUTLINK\_RESTART* bit (these two bits are cleared automatically by hardware). By doing so, hardware can obtain the address of the first descriptor in the new list when reading the last descriptor in the loaded list, and then read the new list.

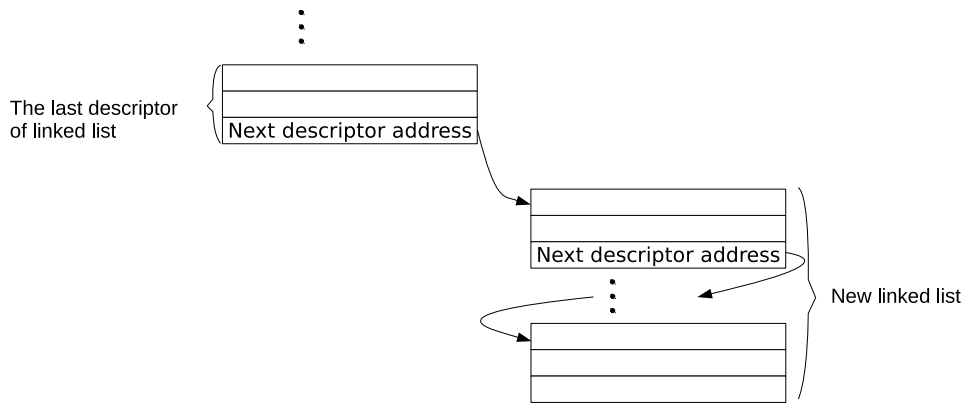


Figure 7-4. Relationship among Linked Lists

### 7.3.4 Linked List reading process

Once configured and enabled by software, the DMA controller starts to read the linked list from internal RAM. `PERI_IN_DSCR_ERR_INT_ENA` bit or `PERI_OUT_DSCR_ERR_INT_ENA` bit can be set to enable descriptor error interrupt. If the buffer address pointer (DW1) does not point to `0x3FFB0000 ~ 0x3FFFFFFF` when the DMA controller accesses internal RAM, or does not point to `0x3F500000 ~ 0x3FF7FFFF` when the DMA controller accesses external RAM, a descriptor error interrupt is generated.

**Note:** The third word (DW2) in a descriptor can only point to internal RAM; it points to the next descriptor to use and descriptors must be in internal memory.

### 7.3.5 EOF

The DMA controller uses EOF (end of file) flags to indicate the completion of data transfer.

Before the DMA controller transmits data, `PERI_OUT_TOTAL_EOF_INT_ENA` bit should be set. If data in the buffer pointed by the last descriptor has been transmitted, a `PERI_OUT_TOTAL_EOF_INT` interrupt is generated.

Before the DMA controller receives data, `PERI_IN_SUC_EOF_INT_ENA` bit should be set. If data has been received successfully, a `PERI_IN_SUC_EOF_INT` interrupt is generated. In addition to `PERI_IN_SUC_EOF_INT` interrupt, UART DMA also supports `UHCI_IN_ERR_EOF_INT`. This interrupt is enabled by setting `UHCI_IN_ERR_EOF_INT_ENA` bit, and it indicates that a data packet has been received with errors.

When a `PERI_OUT_TOTAL_EOF_INT` or a `PERI_IN_SUC_EOF_INT` interrupt is detected, software can record the value of field `PERI_OUTLINK_DSCR_ADDR` or `PERI_INLINK_DSCR_ADDR`, i.e. address of the last descriptor (right shifted 2 bits). Therefore, software can tell which descriptors have been used and reclaim them.

**Note:** In this chapter, EOF of transmit descriptors refers to `suc_eof`, EOF of receive descriptors refers to both `suc_eof` and `err_eof`.

### 7.3.6 Internal DMA

Internal DMA is used by UART0, UART1, SPI3, and ADC Controller. It can only access `0x3FFB0000 ~ 0x3FFFFFFF` in internal RAM. For Internal DMA, size, length, and buffer address pointer in linked list descriptors are not necessarily word-aligned. In other words, Internal DMA can read data of specified length from any starting address in the accessible address range, or write data of the specified length to any contiguous addresses in the accessible address range.

### 7.3.7 EDMA

EDMA is used by I2S0, SPI2, AES Accelerator, and SHA Accelerator. It can access both internal RAM and external RAM.

#### 7.3.7.1 Accessing Internal RAM

Like Internal DMA, EDMA can access 0x3FFB0000 ~ 0x3FFFFFFF in internal RAM. Size, length, and buffer address pointer in linked list descriptors are also not necessarily word-aligned. Please note that if EDMA receives more than one data frame with null character (null character indicates empty data, which will not be written into memory by EDMA; such a character is generated together with EOF), as shown in Figure 7-5 there must be at least three bytes between EOF in the first data frame and EOF in the second data frame. In this case, EOF could be either suc\_eof or err\_eof.

EDMA can send data in burst mode to improve data transfer efficiency. To enable burst mode, please set `PERI_OUT_DATA_BURST_EN` bit.

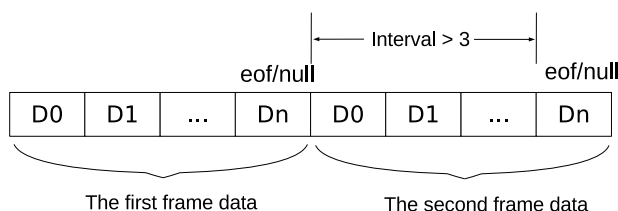


Figure 7-5. EDMA Receiving Data Frames in Internal

### 7.3.8 Accessing External RAM

EDMA can access 0x3F500000 ~ 0x3FF7FFFF in external RAM. The block size of EDMA can be 16 bytes, 32 bytes or 64 bytes. That is, the amount of data to transfer at a time can be 16 bytes, 32 bytes or 64 bytes. To configure the block size, please set `PERI_EXT_MEM_BK_SIZE` bit. Note that all EDMA should have the same block size.

When EDMA accesses external RAM, fields in linked list descriptors should be aligned with block size. Specifically, size and buffer address pointer in receive descriptors should be 16-byte, 32-byte or 64-byte aligned. For data frame whose length is not a multiple of 16 bytes, 32 bytes, or 64 bytes, EDMA adds padding bytes to the end. After `PERI_DMA_IN_SUC_EOF_INT` is detected, software can check the amount of valid data by reading length field in receive descriptors. Size, length and buffer address pointer in transmit descriptors are not necessarily aligned with block size. Table 36 illustrates the value of `PERI_EXT_MEM_BK_SIZE` bit when fields in linked list descriptors are 16-byte, 32-byte and 64-byte aligned respectively.

**Table 36: Relationship Between Configuration Register, Block Size and Alignment**

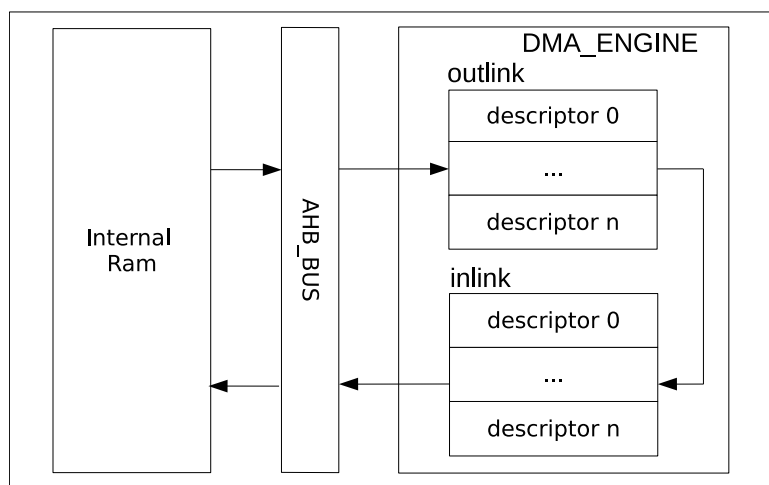
<i>PERI_EXT_MEM_BK_SIZE</i>	Block Size	Alignment
0	16 bytes	16-byte aligned
1	32 bytes	32-byte aligned
2	64 bytes	64-byte aligned

## 7.4 Copy DMA Controller

Copy DMA is used for data transfer from one location in internal RAM to another. Figure 7-6 shows the architecture of a Copy DMA engine. Unlike Internal DMA and EDMA, Copy DMA first reads data to be transferred from internal RAM, stores the data into the DMA FIFO via an outlink, and then writes the data to the target internal RAM via an inlink.

Copy DMA should be configured by software as follows:

1. Set `CP_DMA_IN_RST`, `CP_DMA_OUT_RST`, `CP_DMA_FIFO_RST` and `CP_DMA_CMDFIFO_RST` bit first to 1 and then to 0, to reset Copy DMA state machine and FIFO pointer;
2. Load an outlink, and configure `CP_DMA_OUTLINK_ADDR` with address of the first transmit descriptor;
3. Load an inlink, and configure `CP_DMA_INLINK_ADDR` with address of the first receive descriptor;
4. Set `CP_DMA_OUTLINK_START` to enable DMA transmission;
5. Set `CP_DMA_INLINK_START` to enable DMA reception.

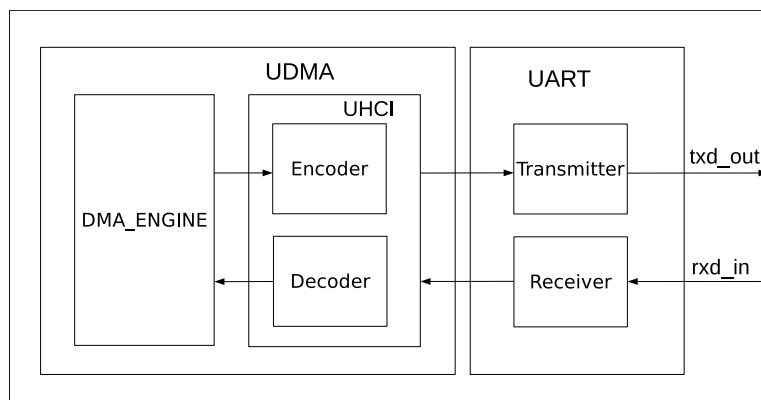
**Figure 7-6. Copy DMA Engine Architecture**

## 7.5 UART DMA (UDMA) Controller

ESP32-S2 has two UART controllers. They share one UDMA controller. `UHCI_UART_CE` specifies which UART controller gets access to UDMA.

Figure 7-7 shows how data is transferred using UDMA. Before UDMA receives data, software prepares an inlink. `UHCI_INLINK_ADDR` points to the first receive descriptor in the inlink. After `UHCI_INLINK_START` is set, UHCI sends data that UART has received to the Decoder. The decoded data is then stored into the RAM pointed by the inlink under the control of UDMA.



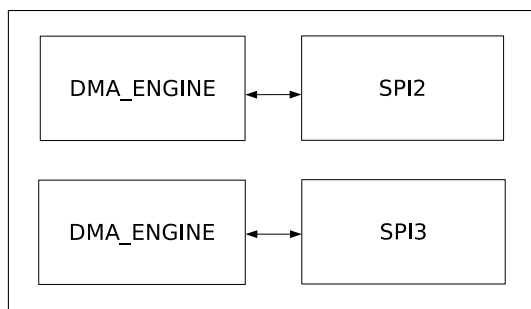


**Figure 7-7. Data Transfer in UDMA Mode**

Before UDMA sends data, software prepares an outlink and data to be sent. [UHCI\\_OUTLINK\\_ADDR](#) points to the first transmit descriptor in the outlink. After [UHCI\\_OUTLINK\\_START](#) is set, UDMA reads data from the RAM pointed by outlink. The data is then encoded by the Encoder, and sent sequentially by the UART transmitter.

Data packets of UDMA have separators at the beginning and the end, with data bits in the middle. The encoder inserts separators in front of and after data bits, and replaces data bits identical to separators with special characters. The decoder removes separators in front of and after data bits, and replaces special characters with separators. There can be more than one continuous separator at the beginning and the end of a data packet. The separator is configured by [UHCI\\_SEPER\\_CHAR](#), 0xC0 by default. The special character is configured by [UHCI\\_ESC\\_SEQ0\\_CHAR0](#) (0xDB by default) and [UHCI\\_ESC\\_SEQ0\\_CHAR1](#) (0xDD by default). When all data has been sent, a UHCI\_OUT\_TOTAL\_EOF\_INT interrupt is generated. When all data has been received, a UHCI\_IN\_SUC\_EOF\_INT is generated.

## 7.6 SPI DMA Controller



**Figure 7-8. SPI DMA**

As shown in Figure 7-8, SPI2 and SPI3 have separate DMA controllers.

SPI DMA receives and transmits data through descriptors at least one byte at a time. The transmission of data can be done in bursts.

SPI\_OUTLINK\_START bit of SPI\_DMA\_OUT\_LINK\_REG register and SPI\_INLINK\_START bit of SPI\_DMA\_IN\_LINK\_REG register are used to enable the DMA engine and are cleared by hardware. When SPI\_OUTLINK\_START is set to 1, the DMA engine loads an outlink and prepares data to be transferred; when SPI\_INLINK\_START is set to 1, the DMA engine loads an inlink and prepares to receive data.

When receiving data, SPI DMA should be configured by software as follows:

1. Set SPI\_DMA\_IN\_RST, SPI\_AHBM\_FIFO\_RST and SPI\_AHBM\_RST bit first to 1 and then to 0, to reset DMA state machine and FIFO pointer;
2. Load an inlink, and configure SPI\_DMA\_INLINK\_ADDR with address of the first receive descriptor;
3. Set SPI\_DMA\_INLINK\_START to enable DMA reception.

When transmitting data, SPI DMA should be configured by software as follows:

1. Set SPI\_DMA\_OUT\_RST, SPI\_AHBM\_FIFO\_RST and SPI\_AHBM\_RST bit first to 1 and then to 0, to reset DMA state machine and FIFO pointer;
2. Load an outlink, and configure SPI\_DMA\_OUTLINK\_ADDR with address of the first transmit descriptor;
3. Set SPI\_DMA\_OUTLINK\_START to enable DMA transmission.

**Note:** When SPI DMA transfers data between internal RAM and external RAM, SPI\_MEM\_TRANS\_EN should be set.

## 7.7 I2S DMA Controller

ESP32-S2 I2S has an individual DMA. I2S\_DSCR\_EN bit of I2S\_FIFO\_CONF\_REG register is used to enable DMA transfer of I2S. I2S DMA receives and transmits data through linked lists. The transmission of data can be done in bursts. I2S\_RX\_EOF\_NUM[31:0] bit of I2S\_RXEOF\_NUM\_REG register is used to configure how many words of data to be received at a time.

I2S\_OUTLINK\_START bit of I2S\_OUT\_LINK\_REG and I2S\_INLINK\_START bit of I2S\_IN\_LINK\_REG register are used to enable the DMA engine and are cleared by hardware. When I2S\_OUTLINK\_START bit is set to 1, the DMA engine loads an outlink and prepares data to be transferred; when I2S\_INLINK\_START is set to 1, the DMA engine loads an inlink and prepares to receive data.

When receiving data, I2S DMA should be configured by software as follows:

1. Set I2S\_IN\_RST, I2S\_AHBM\_FIFO\_RST and I2S\_AHBM\_RST bit first to 1 and then to 0, to reset DMA state machine and FIFO pointer;
2. Load an inlink, and configure I2S\_INLINK\_ADDR with address of the first receive descriptor;
3. Set I2S\_INLINK\_START to enable DMA reception.

When transmitting data, I2S DMA should be configured by software as follows:

1. Set I2S\_OUT\_RST, I2S\_AHBM\_FIFO\_RST and I2S\_AHBM\_RST bit first to 1 and then to 0, to reset DMA state machine and FIFO pointer;
2. Load an outlink, and configure I2S\_OUTLINK\_ADDR with address of the first transmit descriptor;
3. Set I2S\_OUTLINK\_START to enable DMA transmission.

**Note:** When I2S DMA transfers data between internal RAM and external RAM using I2S DMA, I2S\_MEM\_TRANS\_EN should be set.

## 7.8 Crypto DMA

Crypto DMA on ESP32-S2 is shared by AES Accelerator and SHA Accelerator. If

[CRYPTO\\_DMA\\_AES\\_SHA\\_SELECT](#) is set, Crypto DMA is used by SHA Accelerator. If

[CRYPTO\\_DMA\\_AES\\_SHA\\_SELECT](#) is cleared, Crypto DMA is used by AES Accelerator.

When receiving data, Crypto DMA should be configured by software as follows:

1. Set [CRYPTO\\_DMA\\_IN\\_RST](#), [CRYPTO\\_DMA\\_AHBM\\_FIFO\\_RST](#) and [CRYPTO\\_DMA\\_AHBM\\_RST](#) bit first to 1 and then to 0, to reset DMA state machine and FIFO pointer;
2. Load an inlink, and configure [CRYPTO\\_DMA\\_INLINK\\_ADDR](#) with address of the first receive descriptor;
3. Set [CRYPTO\\_DMA\\_INLINK\\_START](#) to enable DMA reception.

When transmitting data, Crypto DMA should be configured by software as follows:

1. Set [CRYPTO\\_DMA\\_OUT\\_RST](#), [CRYPTO\\_DMA\\_AHBM\\_FIFO\\_RST](#) and [CRYPTO\\_DMA\\_AHBM\\_RST](#) first to 1 and then to 0, to reset DMA state machine and FIFO pointer;
2. Load an outlink, and configure [CRYPTO\\_DMA\\_OUTLINK\\_ADDR](#) with address of the first transmit descriptor;
3. Set [CRYPTO\\_DMA\\_OUTLINK\\_START](#) to enable DMA transmission.

**Note:** When Crypto DMA is used for data transfer between internal RAM and external RAM, please set [CRYPTO\\_DMA\\_MEM\\_TRANS\\_EN](#).

## 7.9 Copy DMA Interrupts

- [CP\\_DMA\\_OUT\\_TOTAL\\_EOF\\_INT](#): Triggered when all data corresponding to a linked list (including multiple descriptors) has been sent.
- [CP\\_DMA\\_IN\\_DSCR\\_EMPTY\\_INT](#): Triggered when the size of the buffer pointed by receive descriptors is smaller than the length of data to be received.
- [CP\\_DMA\\_OUT\\_DSCR\\_ERR\\_INT](#): Triggered when an error is detected in a transmit descriptor.
- [CP\\_DMA\\_IN\\_DSCR\\_ERR\\_INT](#): Triggered when an error is detected in a receive descriptor.
- [CP\\_DMA\\_OUT\\_EOF\\_INT](#): Triggered when EOF in a transmit descriptor is 1 and data corresponding to this descriptor has been sent.
- [CP\\_DMA\\_OUT\\_DONE\\_INT](#): Triggered when all data corresponding to a transmit descriptor has been sent.
- [CP\\_DMA\\_IN\\_SUC\\_EOF\\_INT](#): Triggered when a data packet has been received.
- [CP\\_DMA\\_IN\\_DONE\\_INT](#): Triggered when all data corresponding to a receive descriptor has been sent.

## 7.10 Crypto DMA Interrupts

- [CRYPTO\\_DMA\\_INFIFO\\_FULL\\_WM\\_INT](#): Triggered when the amount of data in DMA RX FIFO is larger than the threshold (the value of [CRYPTO\\_DMA\\_INFIFO\\_FULL\\_THRS](#)).
- [CRYPTO\\_DMA\\_OUT\\_TOTAL\\_EOF\\_INT](#): Triggered when all data corresponding to a linked list (including multiple descriptors) has been sent.

- CRYPTO\_DMA\_IN\_DSCR\_EMPTY\_INT: Triggered when the size of the buffer pointed by receive descriptors is smaller than the length of data to be received.
- CRYPTO\_DMA\_OUT\_DSCR\_ERR\_INT: Triggered when an error is detected in a receive descriptor.
- CRYPTO\_DMA\_IN\_DSCR\_ERR\_INT: Triggered when an error is detected in a transmit descriptor.
- CRYPTO\_DMA\_OUT\_EOF\_INT: Triggered when EOF in a transmit descriptor is 1 and all data corresponding to this descriptor has been sent.
- CRYPTO\_DMA\_OUT\_DONE\_INT: Triggered when all data corresponding to a transmit descriptor has been sent.
- CRYPTO\_DMA\_IN\_ERR\_EOF\_INT: Reserved.
- CRYPTO\_DMA\_IN\_SUC\_EOF\_INT: Triggered when a data packet has been received.
- CRYPTO\_DMA\_IN\_DONE\_INT: Triggered when all data corresponding to a receive descriptor has been sent.

## 7.11 Base Address

Users can access Copy DMA and Crypto DMA respectively with two base addresses, which can be seen in table 37. Details about UART DMA, SPI DMA and I2S DMA are included in individual chapters. For more information about accessing peripherals from different buses, please see Chapter 1 *System and Memory*.

**Table 37: Copy DMA and Crypto DMA Base Address**

Module	Bus to Access Peripheral	Base Address
CP DMA	PeriBUS1	0x3F4C3000
CRYPTO DMA	PeriBUS1	0x3F43F000
	PeriBUS2	0x6003F000

## 7.12 Register Summary

The addresses in the following table are relative to the Copy DMA base addresses provided in Section 7.11.

Name	Description	Address	Access
<b>Interrupt Registers</b>			
<a href="#">CP_DMA_INT_RAW_REG</a>	Raw interrupt status	0x0000	RO
<a href="#">CP_DMA_INT_ST_REG</a>	Masked interrupt status	0x0004	RO
<a href="#">CP_DMA_INT_ENA_REG</a>	Interrupt enable bits	0x0008	R/W
<a href="#">CP_DMA_INT_CLR_REG</a>	Interrupt clear bits	0x000C	WO
<b>Configuration Registers</b>			
<a href="#">CP_DMA_OUT_LINK_REG</a>	Link descriptor address and control	0x0010	varies
<a href="#">CP_DMA_IN_LINK_REG</a>	Link descriptor address and control	0x0014	varies
<a href="#">CP_DMA_CONF_REG</a>	Copy DMA configuration register	0x003C	R/W
<b>Status Registers</b>			
<a href="#">CP_DMA_OUT_EOF_DES_ADDR_REG</a>	Transmit descriptor address when EOF occurs	0x0018	RO
<a href="#">CP_DMA_IN_EOF_DES_ADDR_REG</a>	Receive descriptor address when EOF occurs	0x001C	RO

Name	Description	Address	Access
CP_DMA_OUT_EOF_BFR_DES_ADDR_REG	Transmit descriptor address before the last transmit descriptor	0x0020	RO
CP_DMA_INLINK_DSCR_REG	Address of current receive descriptor	0x0024	RO
CP_DMA_INLINK_DSCR_BF0_REG	Address of last receive descriptor	0x0028	RO
CP_DMA_OUTLINK_DSCR_REG	Address of current transmit descriptor	0x0030	RO
CP_DMA_OUTLINK_DSCR_BF0_REG	Address of last transmit descriptor	0x0034	RO
CP_DMA_IN_ST_REG	Status register of receiving data	0x0040	RO
CP_DMA_OUT_ST_REG	Status register of transmitting data	0x0044	RO
CP_DMA_DATE_REG	Copy DMA version control register	0x00FC	R/W

The addresses in the following table are relative to the Crypto DMA base addresses provided in Section 7.11.

Name	Description	Address	Access
<b>Configuration Registers</b>			
CRYPTO_DMA_CONF0_REG	DMA configuration register	0x0000	R/W
CRYPTO_DMA_OUT_LINK_REG	Link descriptor address and control	0x0024	varies
CRYPTO_DMA_IN_LINK_REG	Link descriptor address and control	0x0028	varies
CRYPTO_DMA_CONF1_REG	DMA configuration register	0x002C	R/W
CRYPTO_DMA_AHB_TEST_REG	AHB test register	0x0048	R/W
CRYPTO_DMA_AES_SHA_SELECT_REG	AES/SHA select register	0x0064	R/W
CRYPTO_DMA_PD_CONF_REG	Power control register	0x0068	R/W
CRYPTO_DMA_DATE_REG	Crypto DMA version control register	0x00FC	R/W
<b>Interrupt Registers</b>			
CRYPTO_DMA_INT_RAW_REG	Raw interrupt status	0x0004	RO
CRYPTO_DMA_INT_ST_REG	Masked interrupt status	0x0008	RO
CRYPTO_DMA_INT_ENA_REG	Interrupt enable bits	0x000C	R/W
CRYPTO_DMA_INT_CLR_REG	Interrupt clear bits	0x0010	WO
<b>Status Registers</b>			
CRYPTO_DMA_OUT_STATUS_REG	TX FIFO status register	0x0014	RO
CRYPTO_DMA_IN_STATUS_REG	RX FIFO status register	0x001C	RO
CRYPTO_DMA_STATE0_REG	Status register of receiving data	0x0030	RO
CRYPTO_DMA_STATE1_REG	Status register of transmitting data	0x0034	RO
CRYPTO_DMA_OUT_EOF_DES_ADDR_REG	Transmit descriptor address when EOF occurs	0x0038	RO
CRYPTO_DMA_IN_SUC_EOF_DES_ADDR_REG	Receive descriptor address when EOF occurs	0x003C	RO
CRYPTO_DMA_IN_ERR_EOF_DES_ADDR_REG	Receive descriptor address when errors occur	0x0040	RO
CRYPTO_DMA_OUT_EOF_BFR_DES_ADDR_REG	Transmit descriptor address before the last transmit descriptor	0x0044	RO
CRYPTO_DMA_IN_DSCR_REG	Address of current receive descriptor	0x004C	RO
CRYPTO_DMA_IN_DSCR_BF0_REG	Address of last receive descriptor	0x0050	RO

Name	Description	Address	Access
<a href="#">CRYPTO_DMA_OUT_DSCR_REG</a>	Address of current transmit descriptor	0x0058	RO
<a href="#">CRYPTO_DMA_OUT_DSCR_BF0_REG</a>	Address of last transmit descriptor	0x005C	RO

## 7.13 Registers

Register 7.1: CP\_DMA\_INT\_RAW\_REG (0x0000)

(reserved)																								CP_DMA_OUT_TOTAL_EOF_INT_RAW CP_DMA_IN_DSCR_EMPTY_INT_RAW CP_DMA_OUT_DSCR_ERR_INT_RAW CP_DMA_IN_DSCR_ERR_INT_RAW CP_DMA_OUT_EOF_INT_RAW CP_DMA_IN_DONE_INT_RAW CP_DMA_IN_SUC_EOF_INT_RAW CP_DMA_IN_DONE_INT_RAW									
31																								8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset			

**CP\_DMA\_IN\_DONE\_INT\_RAW** This is the interrupt raw bit. Triggered when the last data of frame is received or the receive buffer is full indicated by receive descriptor. (RO)

**CP\_DMA\_IN\_SUC\_EOF\_INT\_RAW** This is the interrupt raw bit. Triggered when the last data of one frame is received. (RO)

**CP\_DMA\_OUT\_DONE\_INT\_RAW** This is the interrupt raw bit. Triggered when all data indicated by one transmit descriptor has been pushed into TX FIFO. (RO)

**CP\_DMA\_OUT\_EOF\_INT\_RAW** This is the interrupt raw bit. Triggered when the last data with EOF flag has been pushed into TX FIFO. (RO)

**CP\_DMA\_IN\_DSCR\_ERR\_INT\_RAW** This is the interrupt raw bit. Triggered when detecting receive descriptor error, including owner error, the second and third word error of receive descriptor. (RO)

**CP\_DMA\_OUT\_DSCR\_ERR\_INT\_RAW** This is the interrupt raw bit. Triggered when detecting transmit descriptor error, including owner error, the second and third word error of transmit descriptor. (RO)

**CP\_DMA\_IN\_DSCR\_EMPTY\_INT\_RAW** This is the interrupt raw bit. Triggered when receiving data is completed and no more receive descriptor. (RO)

**CP\_DMA\_OUT\_TOTAL\_EOF\_INT\_RAW** This is the interrupt raw bit. Triggered when data corresponding to all transmit descriptors and the last descriptor with valid EOF is transmitted out. (RO)

Register 7.2: CP\_DMA\_INT\_ST\_REG (0x0004)

(reserved)																								CP_DMA_OUT_TOTAL_EOF_INT_ST CP_DMA_IN_DSCR_EMPTY_INT_ST CP_DMA_OUT_DSCR_ERR_INT_ST CP_DMA_IN_DSCR_ERR_INT_ST CP_DMA_OUT_EOF_INT_ST CP_DMA_IN_DONE_INT_ST CP_DMA_IN_SUC_EOF_INT_ST CP_DMA_IN_DONE_INT_ST									
31																								8	7	6	5	4	3	2	1	0	Reset
0 0																								0	0	0	0	0	0	0	0	0	

**CP\_DMA\_IN\_DONE\_INT\_ST** This is the masked interrupt bit for CP\_DMA\_IN\_DONE\_INT interrupt when CP\_DMA\_IN\_DONE\_INT\_ENA is set. (RO)

**CP\_DMA\_IN\_SUC\_EOF\_INT\_ST** This is the masked interrupt bit for CP\_DMA\_IN\_SUC\_EOF\_INT interrupt when CP\_DMA\_IN\_SUC\_EOF\_INT\_ENA is set. (RO)

**CP\_DMA\_OUT\_DONE\_INT\_ST** This is the masked interrupt bit for CP\_DMA\_OUT\_DONE\_INT interrupt when CP\_DMA\_OUT\_DONE\_INT\_ENA is set. (RO)

**CP\_DMA\_OUT\_EOF\_INT\_ST** This is the masked interrupt bit for CP\_DMA\_OUT\_EOF\_INT interrupt when CP\_DMA\_OUT\_EOF\_INT\_ENA is set. (RO)

**CP\_DMA\_IN\_DSCR\_ERR\_INT\_ST** This is the masked interrupt bit for CP\_DMA\_IN\_DSCR\_ERR\_INT interrupt when CP\_DMA\_IN\_DSCR\_ERR\_INT\_ENA is set. (RO)

**CP\_DMA\_OUT\_DSCR\_ERR\_INT\_ST** This is the masked interrupt bit for CP\_DMA\_OUT\_DSCR\_ERR\_INT interrupt when CP\_DMA\_OUT\_DSCR\_ERR\_INT\_ENA is set. (RO)

**CP\_DMA\_IN\_DSCR\_EMPTY\_INT\_ST** This is the masked interrupt bit for CP\_DMA\_IN\_DSCR\_EMPTY\_INT interrupt when CP\_DMA\_IN\_DSCR\_EMPTY\_INT\_ENA is set. (RO)

**CP\_DMA\_OUT\_TOTAL\_EOF\_INT\_ST** This is the masked interrupt bit for CP\_DMA\_OUT\_TOTAL\_EOF\_INT interrupt when CP\_DMA\_OUT\_TOTAL\_EOF\_INT\_ENA is set. (RO)



Register 7.3: CP\_DMA\_INT\_ENA\_REG (0x0008)

(reserved)																CP_DMA_OUT_TOTAL_EOF_INT_ENA			
																CP_DMA_IN_DSCR_EMPTY_INT_ENA			
																CP_DMA_OUT_DSCR_ERR_INT_ENA			
																CP_DMA_IN_DSCR_ERR_INT_ENA			
																CP_DMA_OUT_EOF_INT_ENA			
																CP_DMA_IN_DONE_INT_ENA			
																CP_DMA_IN_SUC_EOF_INT_ENA			
																CP_DMA_OUT_DONE_INT_ENA			
																CP_DMA_IN_DONE_INT_ENA			
31								8	7	6	5	4	3	2	1	0	Reset		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**CP\_DMA\_IN\_DONE\_INT\_ENA** This is the interrupt enable bit for CP\_DMA\_IN\_DONE\_INT interrupt. (R/W)

**CP\_DMA\_IN\_SUC\_EOF\_INT\_ENA** This is the interrupt enable bit for CP\_DMA\_IN\_SUC\_EOF\_INT interrupt. (R/W)

**CP\_DMA\_OUT\_DONE\_INT\_ENA** This is the interrupt enable bit for CP\_DMA\_OUT\_DONE\_INT interrupt. (R/W)

**CP\_DMA\_OUT\_EOF\_INT\_ENA** This is the interrupt enable bit for CP\_DMA\_OUT\_EOF\_INT interrupt. (R/W)

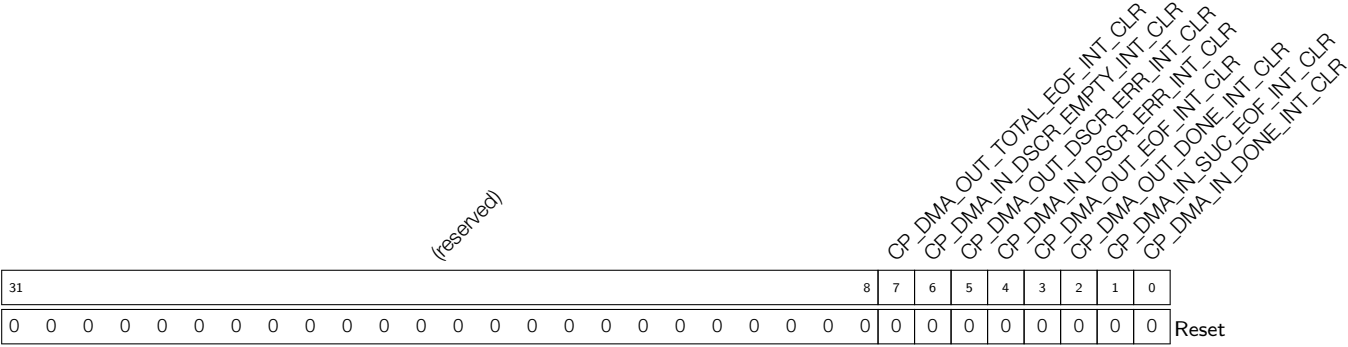
**CP\_DMA\_IN\_DSCR\_ERR\_INT\_ENA** This is the interrupt enable bit for CP\_DMA\_IN\_DSCR\_ERR\_INT interrupt. (R/W)

**CP\_DMA\_OUT\_DSCR\_ERR\_INT\_ENA** This is the interrupt enable bit for CP\_DMA\_OUT\_DSCR\_ERR\_INT interrupt. (R/W)

**CP\_DMA\_IN\_DSCR\_EMPTY\_INT\_ENA** This is the interrupt enable bit for CP\_DMA\_IN\_DSCR\_EMPTY\_INT interrupt. (R/W)

**CP\_DMA\_OUT\_TOTAL\_EOF\_INT\_ENA** This is the interrupt enable bit for CP\_DMA\_OUT\_TOTAL\_EOF\_INT interrupt. (R/W)

Register 7.4: CP\_DMA\_INT\_CLR\_REG (0x000C)



- CP\_DMA\_IN\_DONE\_INT\_CLR** Set this bit to clear CP\_DMA\_IN\_DONE\_INT INTERRUPT. (WO)
- CP\_DMA\_IN\_SUC\_EOF\_INT\_CLR** Set this bit to clear CP\_DMA\_IN\_SUC\_EOF\_INT interrupt. (WO)
- CP\_DMA\_OUT\_DONE\_INT\_CLR** Set this bit to clear CP\_DMA\_OUT\_DONE\_INT interrupt. (WO)
- CP\_DMA\_OUT\_EOF\_INT\_CLR** Set this bit to clear CP\_DMA\_OUT\_EOF\_INT interrupt. (WO)
- CP\_DMA\_IN\_DSCR\_ERR\_INT\_CLR** Set this bit to clear CP\_DMA\_IN\_DSCR\_ERR\_INT interrupt. (WO)
- CP\_DMA\_OUT\_DSCR\_ERR\_INT\_CLR** Set this bit to clear CP\_DMA\_OUT\_DSCR\_ERR\_INT interrupt. (WO)
- CP\_DMA\_IN\_DSCR\_EMPTY\_INT\_CLR** Set this bit to clear CP\_DMA\_IN\_DSCR\_EMPTY\_INT interrupt. (WO)
- CP\_DMA\_OUT\_TOTAL\_EOF\_INT\_CLR** Set this bit to clear CP\_DMA\_OUT\_TOTAL\_EOF\_INT interrupt. (WO)

### Register 7.5: CP\_DMA\_OUT\_LINK\_REG (0x0010)

CP_DMA_OUTLINK_PARK																CP_DMA_OUTLINK_RESTART																CP_DMA_OUTLINK_START																CP_DMA_OUTLINK_STOP																(reserved)																CP_DMA_OUTLINK_ADDR															
31	30	29	28	27												20	19																									0																																																					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x000																								Reset																																																									

**CP\_DMA\_OUTLINK\_ADDR** This register is used to specify the least significant 20 bits of the first transmit descriptor's address. (R/W)

**CP\_DMA\_OUTLINK\_STOP** Set this bit to stop DMA from reading transmit descriptors after finishing the current data transaction. (R/W)

**CP\_DMA\_OUTLINK\_START** Set this bit to start a new transmit descriptor. (R/W)

**CP\_DMA\_OUTLINK\_RESTART** Set this bit to restart the transmit descriptor from the last address.  
(R/W)

**CP\_DMA\_OUTLINK\_PARK** 1: the transmit descriptor's FSM is in idle state. 0: the transmit descriptor's FSM is working. (RO)

### Register 7.6: CP\_DMA\_IN\_LINK\_REG (0x0014)

CP_DMA_INLINK_PARK																															(reserved)																			CP_DMA_INLINK_ADDR																		
CP_DMA_INLINK_RESTART																																																																				
CP_DMA_INLINK_START																																																																				
CP_DMA_INLINK_STOP																																																																				
31	30	29	28	27																	20	19																	0																													
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x000																Reset																																		

**CP\_DMA\_INLINK\_ADDR** This register is used to specify the least significant 20 bits of the first receive descriptor's address. (R/W)

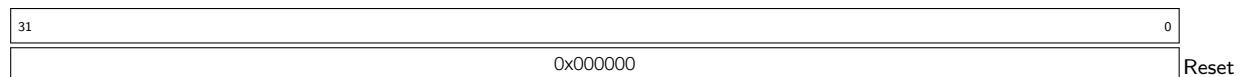
**CP\_DMA\_INLINK\_STOP** Set this bit to stop DMA from reading receive descriptors after finishing the current data transaction. (R/W)

**CP\_DMA\_INLINK\_START** Set this bit to enable DMA to read receive descriptors. (R/W)

**CP\_DMA\_INLINK\_RESTART** Set this bit to restart new receive descriptors. (R/W)

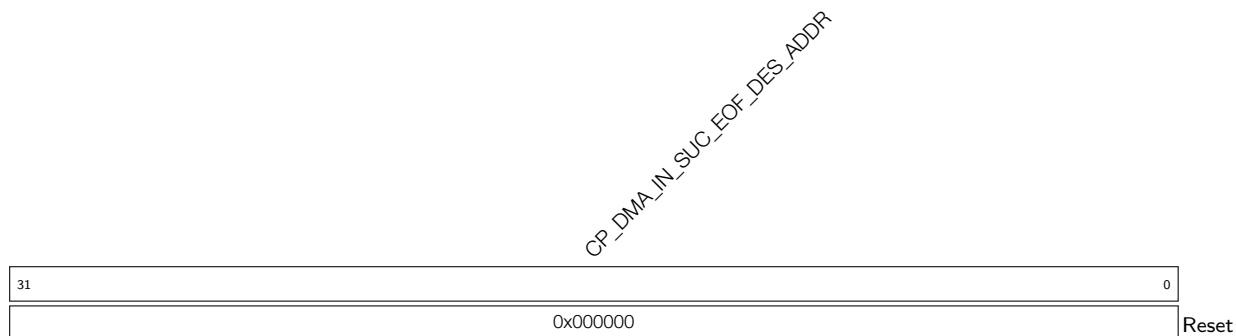
**CP\_DMA\_INLINK\_PARK** 1: the receive descriptor's FSM is in idle state. 0: the receive descriptor's FSM is working. (RO)

[Submit Documentation Feedback](#)

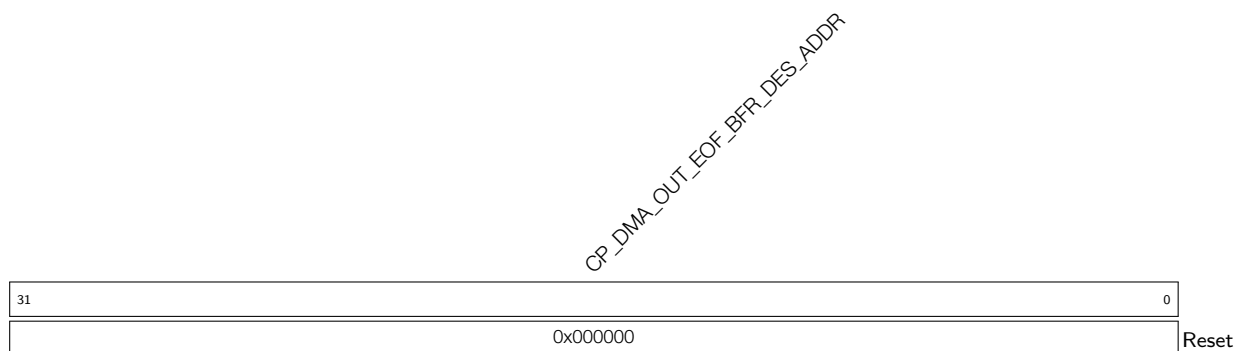


**CP\_DMA\_CLK\_EN** 1'b1: Force clock on for register. 1'b0: Support clock only when application writes registers. (R/W)

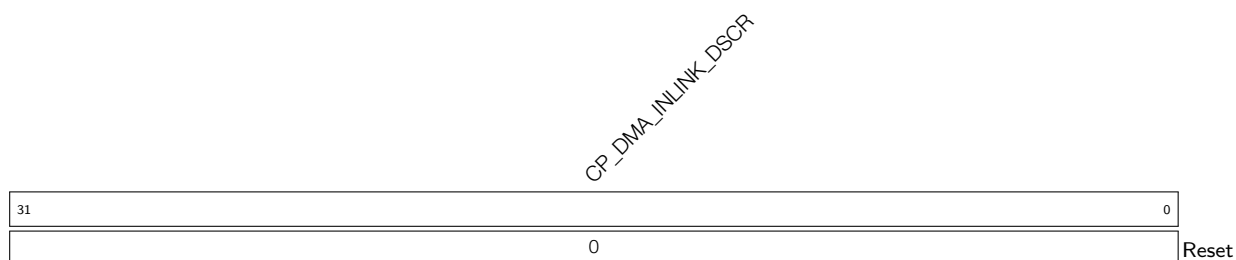
**CP\_DMA\_OUT\_EOF\_DES\_ADDR** This register stores the address of the transmit descriptor when the EOF bit in this descriptor is 1. (RO)

**Register 7.9: CP\_DMA\_IN\_EOF\_DES\_ADDR\_REG (0x001C)**

**CP\_DMA\_IN\_SUC\_EOF\_DES\_ADDR** This register stores the address of the receive descriptor when received successful EOF. (RO)

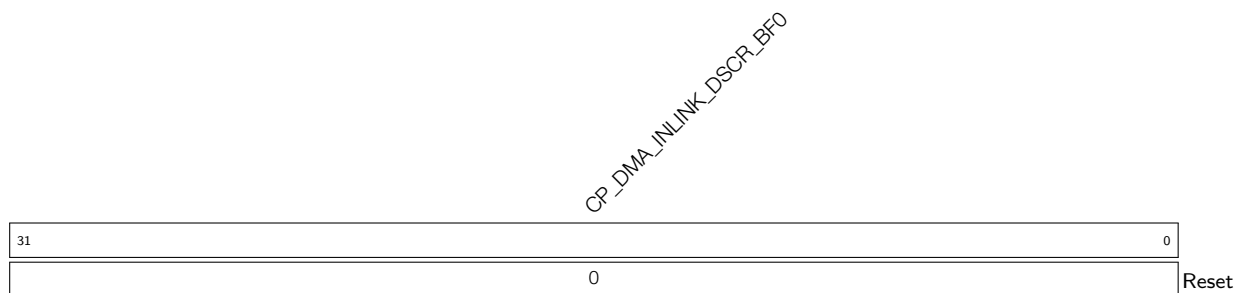
**Register 7.10: CP\_DMA\_OUT\_EOF\_BFR\_DES\_ADDR\_REG (0x0020)**

**CP\_DMA\_OUT\_EOF\_BFR\_DES\_ADDR** This register stores the address of the transmit descriptor before the last transmit descriptor. (RO)

**Register 7.11: CP\_DMA\_INLINK\_DSCR\_REG (0x0024)**

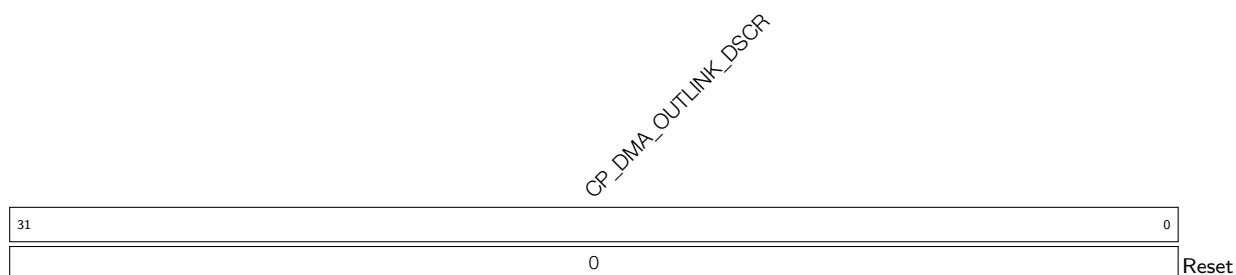
**CP\_DMA\_INLINK\_DSCR** The address of the current receive descriptor x. (RO)

Register 7.12: CP\_DMA\_INLINK\_DSCR\_BF0\_REG (0x0028)



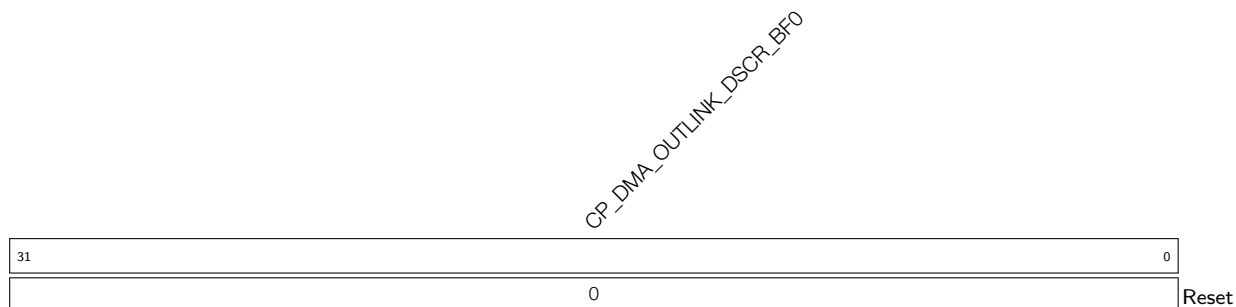
**CP\_DMA\_INLINK\_DSCR\_BF0** The address of the last receive descriptor x-1. (RO)

Register 7.13: CP\_DMA\_OUTLINK\_DSCR\_REG (0x0030)



**CP\_DMA\_OUTLINK\_DSCR** The address of the current transmit descriptor y. (RO)

Register 7.14: CP\_DMA\_OUTLINK\_DSCR\_BF0\_REG (0x0034)



**CP\_DMA\_OUTLINK\_DSCR\_BF0** The address of the last transmit descriptor y-1. (RO)

**Register 7.15: CP\_DMA\_IN\_ST\_REG (0x0040)**

(reserved)																CP_DMA_FIFO_EMPTY				CP_DMA_IN_STATE				CP_DMA_IN_DSCR_STATE				CP_DMA_INLINK_DSCR_ADDR																																			
31								24								23	22				20				19	18	17								0																												
0								0								0								0								0								0								0								Reset							

**CP\_DMA\_INLINK\_DSCR\_ADDR** This register stores the current receive descriptor's address. (RO)

**CP\_DMA\_IN\_DSCR\_STATE** Reserved. (RO)

**CP\_DMA\_IN\_STATE** Reserved. (RO)

**CP\_DMA\_FIFO\_EMPTY** Copy DMA FIFO empty signal. (RO)

**Register 7.16: CP\_DMA\_OUT\_ST\_REG (0x0044)**

(reserved)																CP_DMA_FIFO_FULL				CP_DMA_OUT_STATE				CP_DMA_OUT_DSCR_STATE				CP_DMA_OUTLINK_DSCR_ADDR																
31								24								23	22				20	19	18	17								0												
0								0								0								0				0																Reset

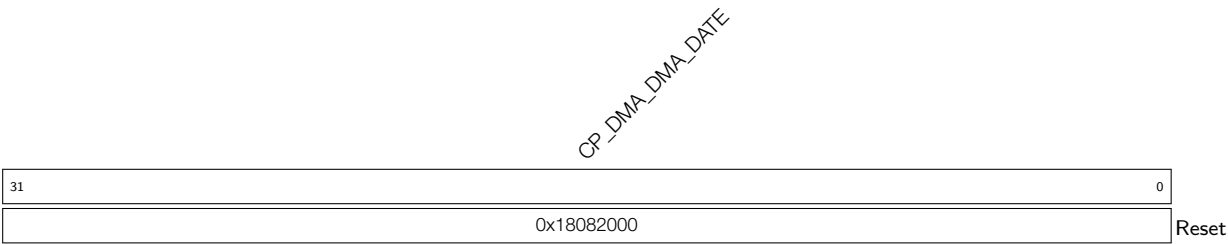
**CP\_DMA\_OUTLINK\_DSCR\_ADDR** This register stores the current transmit descriptor's address. (RO)

**CP\_DMA\_OUT\_DSCR\_STATE** Reserved. (RO)

**CP\_DMA\_OUT\_STATE** Reserved. (RO)

**CP\_DMA\_FIFO\_FULL** Copy DMA FIFO full signal. (RO)

Register 7.17: CP\_DMA\_DATE\_REG (0x00FC)



**CP\_DMA\_DMA\_DATE** This is the version control register. (R/W)



Register 7.18: CRYPTO\_DMA\_CONF0\_REG (0x0000)

(reserved)																				CRYPTO_DMA_MEM_TRANS_EN CRYPTO_DMA_OUT_DATA_BURST_EN CRYPTO_DMA_INDSCR_BURST_EN CRYPTO_DMA_OUTDSCR_BURST_EN CRYPTO_DMA_OUT_EOF_MODE CRYPTO_DMA_OUT_NO_RESTART_CLR CRYPTO_DMA_OUT_AUTO_WBACK CRYPTO_DMA_IN_LOOP_TEST CRYPTO_DMA_AHB_RST CRYPTO_DMA_AHB_FIFO_RST CRYPTO_DMA_OUT_RST CRYPTO_DMA_IN_RST													
31													13							12	11	10	9	8	7	6	5	4	3	2	1	0	Reset
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0													0	0	0	0	1	0	0	0	0	0	0	0	0	0							

**CRYPTO\_DMA\_IN\_RST** This bit is used to reset crypto DMA in FSM and RX FIFO pointer. (R/W)

**CRYPTO\_DMA\_OUT\_RST** This bit is used to reset crypto DMA out FSM and TX FIFO pointer. (R/W)

**CRYPTO\_DMA\_AHB\_FIFO\_RST** This bit is used to reset crypto DMA AHB master FIFO pointer. (R/W)

**CRYPTO\_DMA\_AHB\_RST** Reset crypto DMA AHB master. (R/W)

**CRYPTO\_DMA\_IN\_LOOP\_TEST** Reserved (R/W)

**CRYPTO\_DMA\_OUT\_LOOP\_TEST** Reserved (R/W)

**CRYPTO\_DMA\_OUT\_AUTO\_WBACK** Set this bit to enable automatic outlink-writeback when all the data in TX Buffer has been transmitted. (R/W)

**CRYPTO\_DMA\_OUT\_NO\_RESTART\_CLR** Reserved (R/W)

**CRYPTO\_DMA\_OUT\_EOF\_MODE** Out EOF flag generation mode of TX FIFO. 1: EOF flag of TX is generated when the last data with EOF would be transmitted has been popped from FIFO of Crypto DMA; 0: EOF flag is generated when the last data with EOF would be transmitted has been pushed into FIFO of Crypto DMA. (R/W)

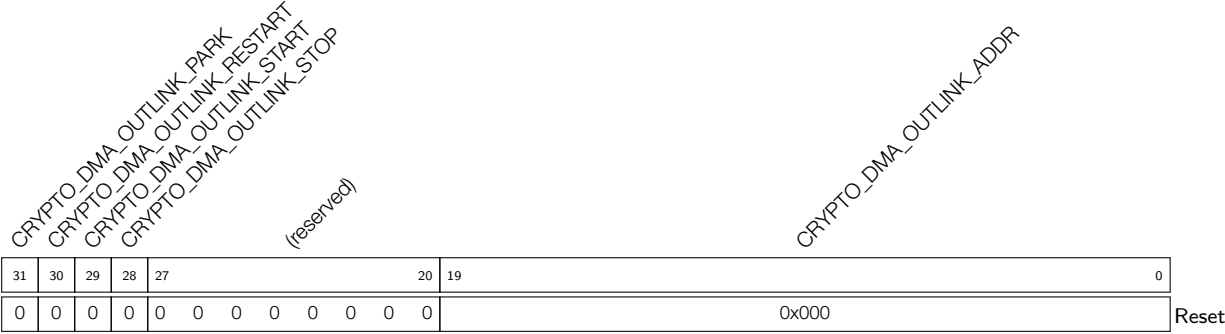
**CRYPTO\_DMA\_OUTDSCR\_BURST\_EN** Set this bit to enable INCR burst transfer when TX FIFO reads descriptor from internal RAM. (R/W)

**CRYPTO\_DMA\_INDSCR\_BURST\_EN** Set this bit to enable INCR burst transfer when RX FIFO reads descriptor from internal RAM. (R/W)

**CRYPTO\_DMA\_OUT\_DATA\_BURST\_EN** Set this bit to enable INCR burst transfer when TX FIFO reads data from internal RAM. (R/W)

**CRYPTO\_DMA\_MEM\_TRANS\_EN** Set this bit to enable automatic transmitting data from memory to memory via DMA. (R/W)

Register 7.19: CRYPTO\_DMA\_OUT\_LINK\_REG (0x0024)



- CRYPTO\_DMA\_OUTLINK\_ADDR** This register stores the 20 least significant bits of the first transmit descriptor's address. (R/W)
- CRYPTO\_DMA\_OUTLINK\_STOP** Set this bit to stop DMA from reading transmit descriptors after finishing the current data transaction. (R/W)
- CRYPTO\_DMA\_OUTLINK\_START** Set this bit to enable DMA to read transmit descriptors. (R/W)
- CRYPTO\_DMA\_OUTLINK\_RESTART** Set this bit to restart a new outlink from the last address. (R/W)
- CRYPTO\_DMA\_OUTLINK\_PARK** 1: the transmit descriptor's FSM is in idle state. 0: the transmit descriptor's FSM is working. (RO)

**Register 7.20: CRYPTO\_DMA\_IN\_LINK\_REG (0x0028)**

CRYPTO_DMA_INLINK_PARK																CRYPTO_DMA_INLINK_RESTART																CRYPTO_DMA_INLINK_STOP																(reserved)																CRYPTO_DMA_INLINK_AUTO_RET																CRYPTO_DMA_INLINK_ADDR																															
31	30	29	28	27											21	20	19																	0																																																																													
0	0	0	0	0	0	0	0	0	0	0	0	1	0x000																																Reset																																																																		

Reset

**CRYPTO\_DMA\_INLINK\_ADDR** This register stores the 20 least significant bits of the first receive descriptor's address. (R/W)

**CRYPTO\_DMA\_INLINK\_AUTO\_RET** Reserved (R/W)

**CRYPTO\_DMA\_INLINK\_STOP** Set this bit to stop DMA from reading receive descriptors after finishing the current data transaction. (R/W)

**CRYPTO\_DMA\_INLINK\_START** Set this bit to enable DMA to read receive descriptors. (R/W)

**CRYPTO\_DMA\_INLINK\_RESTART** Set this bit to mount a new receive descriptor. (R/W)

**CRYPTO\_DMA\_INLINK\_PARK** 1: the receive descriptor's FSM is in idle state. 0: the receive descriptor's FSM is working. (RO)

**Register 7.21: CRYPTO\_DMA\_CONF1\_REG (0x002C)**

(reserved)																CRYPTO_DMA_EXT_MEM_BLK_SIZE				CRYPTO_DMA_CHECK_OWNER				CRYPTO_DMA_INFIFO_FULL_THRS														
31																15		14	13	12	11																0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0		0	0																Reset			

Reset

**CRYPTO\_DMA\_INFIFO\_FULL\_THRS** This register is used to generate the CRYPTO\_DMA\_INFIFO\_FULL\_WM\_INT interrupt when the byte number is up to the value of the register. (R/W)

**CRYPTO\_DMA\_CHECK\_OWNER** Set this bit to enable checking the owner attribute of the link descriptor. (R/W)

**CRYPTO\_DMA\_EXT\_MEM\_BLK\_SIZE** DMA access external memory block size. 0: 16 bytes; 1: 32 bytes; 2: 64 bytes; 3: Reserved. (R/W)

Register 7.22: CRYPTO\_DMA\_AHB\_TEST\_REG (0x0048)

(reserved)																								CRYPTO_DMA_AHB_TESTADDR		(reserved)		CRYPTO_DMA_AHB_TESTMODE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
31																							6	5	4	3	2	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CRYPTO\_DMA\_AHB\_TESTMODE    Reserved (R/W)

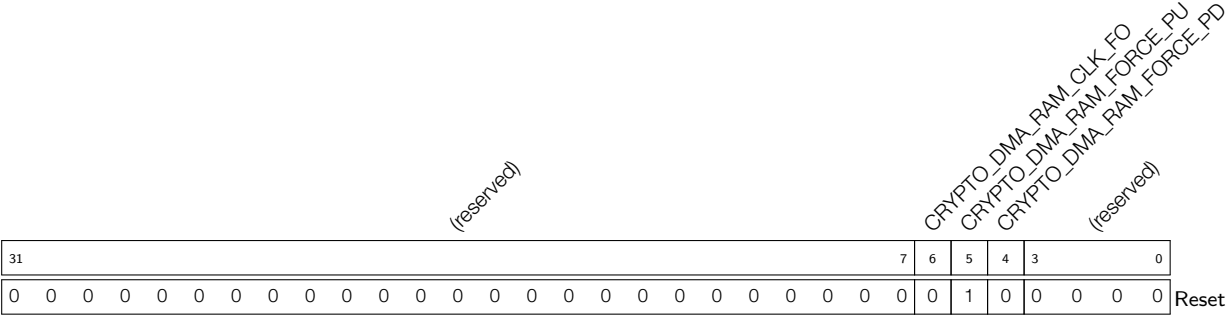
CRYPTO\_DMA\_AHB\_TESTADDR    Reserved (R/W)

Register 7.23: CRYPTO\_DMA\_AES\_SHA\_SELECT\_REG (0x0064)

(reserved)																															CRYPTO_DMA																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CRYPTO\_DMA\_AES\_SHA\_SELECT    Select one between AES and SHA to use DMA. 0: AES. 1:SHA.  
(R/W)

Register 7.24: CRYPTO\_DMA\_PD\_CONF\_REG (0x0068)

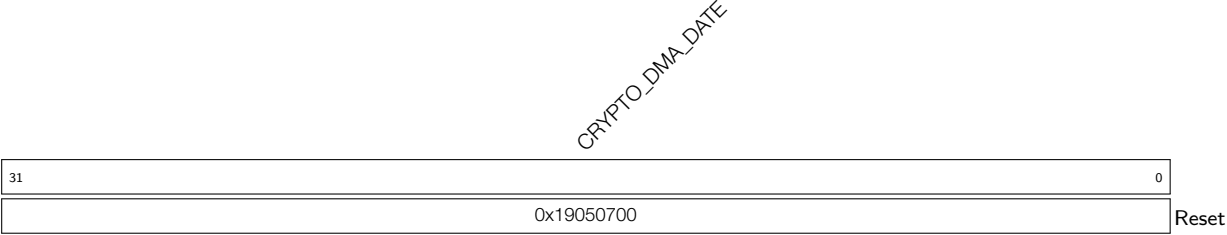


**CRYPTO\_DMA\_RAM\_FORCE\_PD** Force power down signal to RAM. 0: force RAM power up; 1: only when CRYPTO\_DMA\_RAM\_FORCE\_PU is 0, power down RAM. (R/W)

**CRYPTO\_DMA\_RAM\_FORCE\_PU** Force power up signal to RAM. 0: only when CRYPTO\_DMA\_RAM\_FORCE\_PD is 1, power down RAM; 1: force RAM power up. (R/W)

**CRYPTO\_DMA\_RAM\_CLK\_FO** 1: Force to open the clock and bypass the gate-clock when accessing the RAM in DMA. 0: A gate-clock will be used when accessing the RAM in DMA. (R/W)

Register 7.25: CRYPTO\_DMA\_DATE\_REG (0x00FC)



**CRYPTO\_DMA\_DATE** This is the version control register. (R/W)

Register 7.26: CRYPTO\_DMA\_INT\_RAW\_REG (0x0004)

(reserved)										CRYPTO_DMA_INFIFO_FULL_WM_INT_RAW CRYPTO_DMA_OUT_TOTAL_EOF_INT_RAW CRYPTO_DMA_IN_DSCR_EMPTY_INT_RAW CRYPTO_DMA_OUT_DSCR_ERR_INT_RAW CRYPTO_DMA_IN_DSCR_ERR_INT_RAW CRYPTO_DMA_OUT_EOF_INT_RAW CRYPTO_DMA_IN_DONE_INT_RAW CRYPTO_DMA_IN_ERR_EOF_INT_RAW CRYPTO_DMA_IN_SUC_EOF_INT_RAW CRYPTO_DMA_IN_DONE_INT_RAW												
31											10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**CRYPTO\_DMA\_IN\_DONE\_INT\_RAW** The raw interrupt status. Set when the last data of one frame is received or the receive buffer is full indicated by receive descriptor. (RO)

**CRYPTO\_DMA\_IN\_SUC\_EOF\_INT\_RAW** The raw interrupt status. Set when the last data of one frame is received by Crypto DMA RX FIFO. (RO)

**CRYPTO\_DMA\_IN\_ERR\_EOF\_INT\_RAW** Reserved (RO)

**CRYPTO\_DMA\_OUT\_DONE\_INT\_RAW** The raw interrupt status. Set when all data indicated by one transmit descriptor has been pushed into TX FIFO. (RO)

**CRYPTO\_DMA\_OUT\_EOF\_INT\_RAW** The raw interrupt status. Set when Out EOF flag is generated. (RO)

**CRYPTO\_DMA\_IN\_DSCR\_ERR\_INT\_RAW** The raw interrupt status. Set when detecting receive descriptor error, including owner error, the second and third word error of receive descriptor. (RO)

**CRYPTO\_DMA\_OUT\_DSCR\_ERR\_INT\_RAW** The raw interrupt status. Set when detecting transmit descriptor error, including owner error, the second and third word error of transmit descriptor. (RO)

**CRYPTO\_DMA\_IN\_DSCR\_EMPTY\_INT\_RAW** The raw interrupt status. Set when receiving data is completed and no more receive descriptor. (RO)

**CRYPTO\_DMA\_OUT\_TOTAL\_EOF\_INT\_RAW** The raw interrupt status. Set when data corresponding to all transmit descriptor and the last descriptor with valid EOF is transmitted out. (RO)

**CRYPTO\_DMA\_INFIFO\_FULL\_WM\_INT\_RAW** The raw interrupt status. Set when received data byte number is up to threshold configured by CRYPTO\_DMA\_INFIFO\_FULL\_THRS in RX FIFO. (RO)

Register 7.27: CRYPTO\_DMA\_INT\_ST\_REG (0x0008)

(reserved)																						CRYPTO_DMA_INFIFO_FULL_WM_INT_ST CRYPTO_DMA_OUT_TOTAL_EOF_INT_ST CRYPTO_DMA_IN_DSCR_EMPTY_INT_ST CRYPTO_DMA_OUT_DSCR_ERR_INT_ST CRYPTO_DMA_IN_DSCR_ERR_INT_ST CRYPTO_DMA_OUT_DONE_INT_ST CRYPTO_DMA_IN_SUC_EOF_INT_ST CRYPTO_DMA_IN_DONE_INT_ST												
31																						10		9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														

**CRYPTO\_DMA\_IN\_DONE\_INT\_ST** The masked interrupt status bit for the CRYPTO\_DMA\_IN\_DONE\_INT interrupt. (RO)

**CRYPTO\_DMA\_IN\_SUC\_EOF\_INT\_ST** The masked interrupt status bit for the CRYPTO\_DMA\_IN\_SUC\_EOF\_INT interrupt. (RO)

**CRYPTO\_DMA\_IN\_ERR\_EOF\_INT\_ST** Reserved (RO)

**CRYPTO\_DMA\_OUT\_DONE\_INT\_ST** The masked interrupt status bit for the CRYPTO\_DMA\_OUT\_DONE\_INT interrupt. (RO)

**CRYPTO\_DMA\_OUT\_EOF\_INT\_ST** The masked interrupt status bit for the CRYPTO\_DMA\_OUT\_EOF\_INT interrupt. (RO)

**CRYPTO\_DMA\_IN\_DSCR\_ERR\_INT\_ST** The masked interrupt status bit for the CRYPTO\_DMA\_IN\_DSCR\_ERR\_INT interrupt. (RO)

**CRYPTO\_DMA\_OUT\_DSCR\_ERR\_INT\_ST** The masked interrupt status bit for the CRYPTO\_DMA\_OUT\_DSCR\_ERR\_INT interrupt. (RO)

**CRYPTO\_DMA\_IN\_DSCR\_EMPTY\_INT\_ST** The masked interrupt status bit for the CRYPTO\_DMA\_IN\_DSCR\_EMPTY\_INT interrupt. (RO)

**CRYPTO\_DMA\_OUT\_TOTAL\_EOF\_INT\_ST** The masked interrupt status bit for the CRYPTO\_DMA\_OUT\_TOTAL\_EOF\_INT interrupt. (RO)

**CRYPTO\_DMA\_INFIFO\_FULL\_WM\_INT\_ST** The masked interrupt status bit for the CRYPTO\_DMA\_INFIFO\_FULL\_WM\_INT interrupt. (RO)

Register 7.28: CRYPTO\_DMA\_INT\_ENA\_REG (0x000C)

(reserved)																						CRYPTO_DMA_INFIFO_FULL_WM_INT_ENA CRYPTO_DMA_OUT_TOTAL_EOF_INT_ENA CRYPTO_DMA_IN_DSCR_EMPTY_INT_ENA CRYPTO_DMA_OUT_DSCR_ERR_INT_ENA CRYPTO_DMA_IN_DSCR_ERR_INT_ENA CRYPTO_DMA_OUT_DONE_INT_ENA CRYPTO_DMA_IN_SUC_EOF_INT_ENA CRYPTO_DMA_IN_DONE_INT_ENA																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
31																						10		9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**CRYPTO\_DMA\_IN\_DONE\_INT\_ENA** The interrupt enable bit for the CRYPTO\_DMA\_IN\_DONE\_INT interrupt. (R/W)

**CRYPTO\_DMA\_IN\_SUC\_EOF\_INT\_ENA** The interrupt enable bit for the CRYPTO\_DMA\_IN\_SUC\_EOF\_INT interrupt. (R/W)

**CRYPTO\_DMA\_IN\_ERR\_EOF\_INT\_ENA** Reserved (R/W)

**CRYPTO\_DMA\_OUT\_DONE\_INT\_ENA** The interrupt enable bit for the CRYPTO\_DMA\_OUT\_DONE\_INT interrupt. (R/W)

**CRYPTO\_DMA\_OUT\_EOF\_INT\_ENA** The interrupt enable bit for the CRYPTO\_DMA\_OUT\_EOF\_INT interrupt. (R/W)

**CRYPTO\_DMA\_IN\_DSCR\_ERR\_INT\_ENA** The interrupt enable bit for the CRYPTO\_DMA\_IN\_DSCR\_ERR\_INT interrupt. (R/W)

**CRYPTO\_DMA\_OUT\_DSCR\_ERR\_INT\_ENA** The interrupt enable bit for the CRYPTO\_DMA\_OUT\_DSCR\_ERR\_INT interrupt. (R/W)

**CRYPTO\_DMA\_IN\_DSCR\_EMPTY\_INT\_ENA** The interrupt enable bit for the CRYPTO\_DMA\_IN\_DSCR\_EMPTY\_INT interrupt. (R/W)

**CRYPTO\_DMA\_OUT\_TOTAL\_EOF\_INT\_ENA** The interrupt enable bit for the CRYPTO\_DMA\_OUT\_TOTAL\_EOF\_INT interrupt. (R/W)

**CRYPTO\_DMA\_INFIFO\_FULL\_WM\_INT\_ENA** The interrupt enable bit for the CRYPTO\_DMA\_INFIFO\_FULL\_WM\_INT interrupt. (R/W)



Register 7.29: CRYPTO\_DMA\_INT\_CLR\_REG (0x0010)

(reserved)																						CRYPTO_DMA_INFIFO_FULL_WM_INT_CLR									
																						CRYPTO_DMA_OUT_TOTAL_EOF_INT_CLR									
																						CRYPTO_DMA_IN_DSCR_EMPTY_INT_CLR									
																						CRYPTO_DMA_OUT_DSCR_ERR_INT_CLR									
																						CRYPTO_DMA_IN_DSCR_ERR_INT_CLR									
																						CRYPTO_DMA_OUT_EOF_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_ERR_EOF_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									
																						CRYPTO_DMA_IN_DONE_INT_CLR									

**CRYPTO\_DMA\_IN\_DONE\_INT\_CLR** Set this bit to clear the CRYPTO\_DMA\_IN\_DONE\_INT interrupt. (WO)

**CRYPTO\_DMA\_IN\_SUC\_EOF\_INT\_CLR** Set this bit to clear the CRYPTO\_DMA\_IN\_SUC\_EOF\_INT interrupt. (WO)

**CRYPTO\_DMA\_IN\_ERR\_EOF\_INT\_CLR** Reserved (WO)

**CRYPTO\_DMA\_OUT\_DONE\_INT\_CLR** Set this bit to clear the CRYPTO\_DMA\_OUT\_DONE\_INT interrupt. (WO)

**CRYPTO\_DMA\_OUT\_EOF\_INT\_CLR** Set this bit to clear the CRYPTO\_DMA\_OUT\_EOF\_INT interrupt. (WO)

**CRYPTO\_DMA\_IN\_DSCR\_ERR\_INT\_CLR** Set this bit to clear the CRYPTO\_DMA\_IN\_DSCR\_ERR\_INT interrupt. (WO)

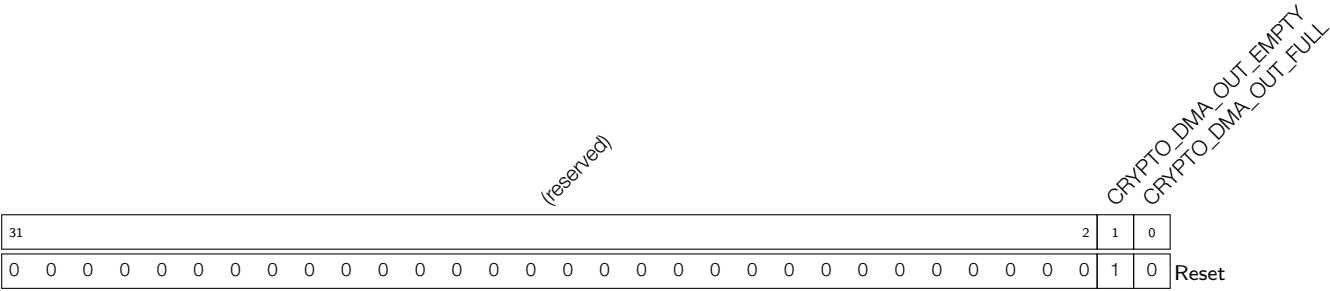
**CRYPTO\_DMA\_OUT\_DSCR\_ERR\_INT\_CLR** Set this bit to clear the CRYPTO\_DMA\_OUT\_DSCR\_ERR\_INT interrupt. (WO)

**CRYPTO\_DMA\_IN\_DSCR\_EMPTY\_INT\_CLR** Set this bit to clear the CRYPTO\_DMA\_IN\_DSCR\_EMPTY\_INT interrupt. (WO)

**CRYPTO\_DMA\_OUT\_TOTAL\_EOF\_INT\_CLR** Set this bit to clear the CRYPTO\_DMA\_OUT\_TOTAL\_EOF\_INT interrupt. (WO)

**CRYPTO\_DMA\_INFIFO\_FULL\_WM\_INT\_CLR** Set this bit to clear the CRYPTO\_DMA\_INFIFO\_FULL\_WM\_INT interrupt. (WO)

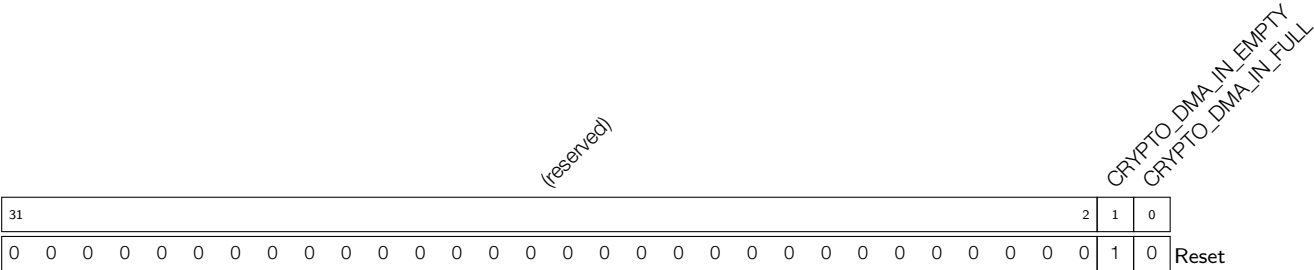
Register 7.30: CRYPTO\_DMA\_OUT\_STATUS\_REG (0x0014)



**CRYPTO\_DMA\_OUT\_FULL** 1: DMA TX FIFO is full. (RO)

**CRYPTO\_DMA\_OUT\_EMPTY** 1: DMA TX FIFO is empty. (RO)

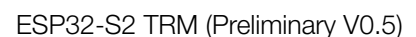
Register 7.31: CRYPTO\_DMA\_IN\_STATUS\_REG (0x001C)

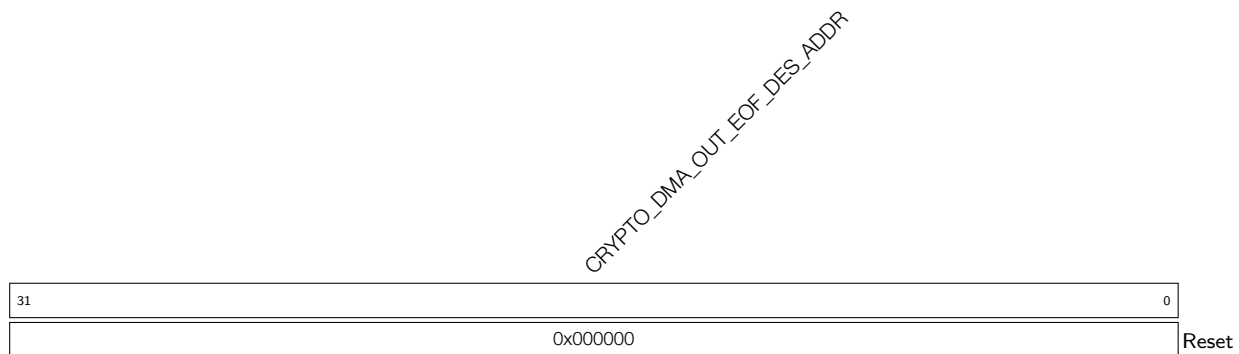


**CRYPTO\_DMA\_IN\_FULL** 1: DMA RX FIFO is full. (RO)

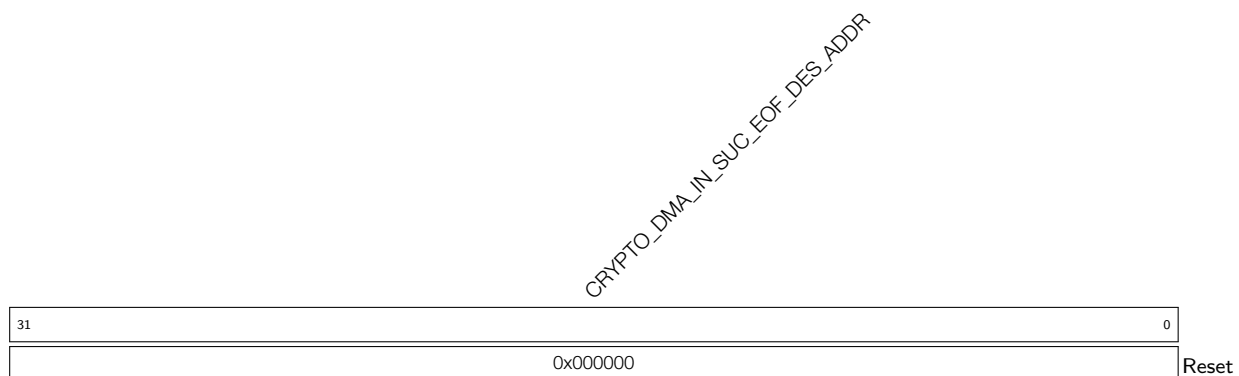
**CRYPTO\_DMA\_IN\_EMPTY** 1: DMA RX FIFO is empty. (RO)

## 203

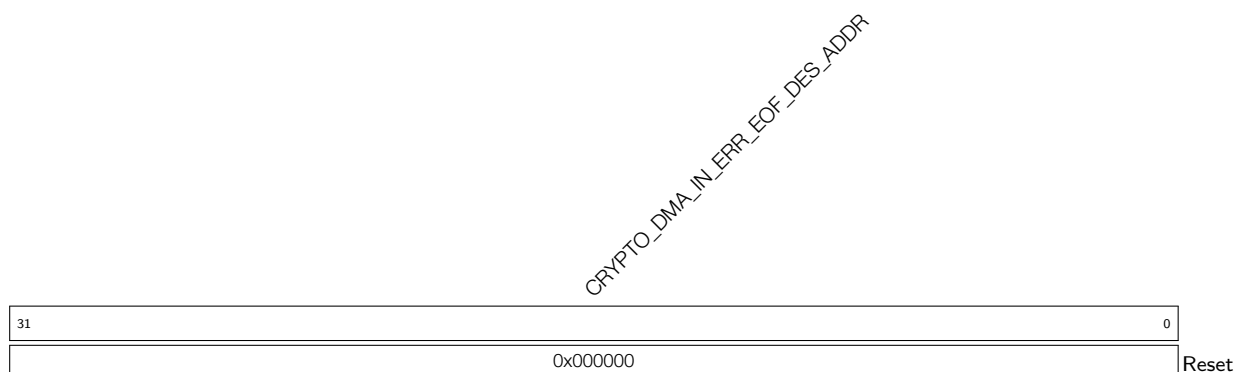


**Register 7.34: CRYPTO\_DMA\_OUT\_EOF\_DES\_ADDR\_REG (0x0038)**

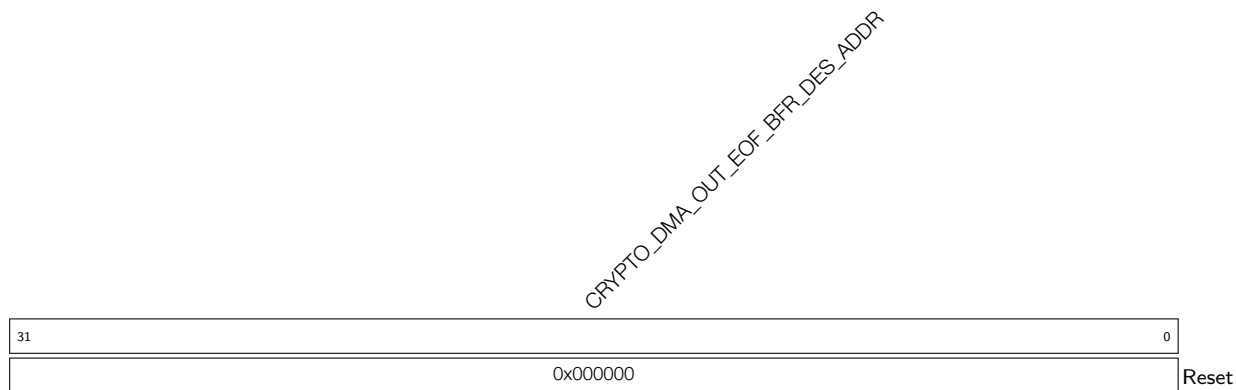
**CRYPTO\_DMA\_OUT\_EOF\_DES\_ADDR** This register stores the address of the transmit descriptor when the EOF bit in this descriptor is 1. (RO)

**Register 7.35: CRYPTO\_DMA\_IN\_SUC\_EOF\_DES\_ADDR\_REG (0x003C)**

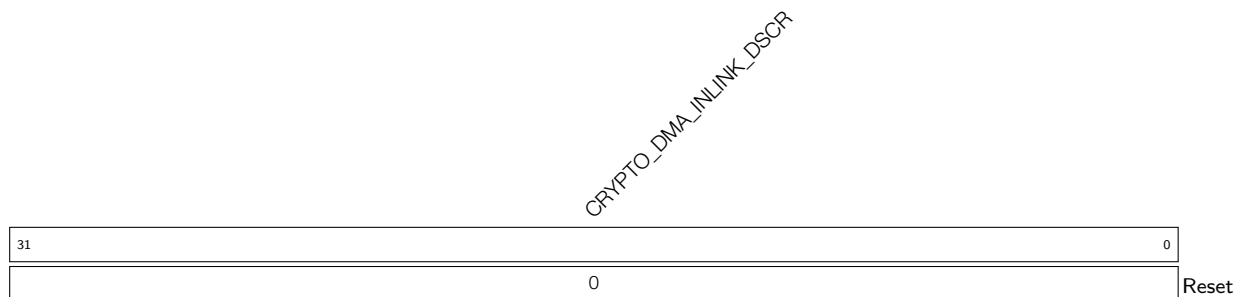
**CRYPTO\_DMA\_IN\_SUC\_EOF\_DES\_ADDR** This register stores the address of the receive descriptor when received successful EOF. (RO)

**Register 7.36: CRYPTO\_DMA\_IN\_ERR\_EOF\_DES\_ADDR\_REG (0x0040)**

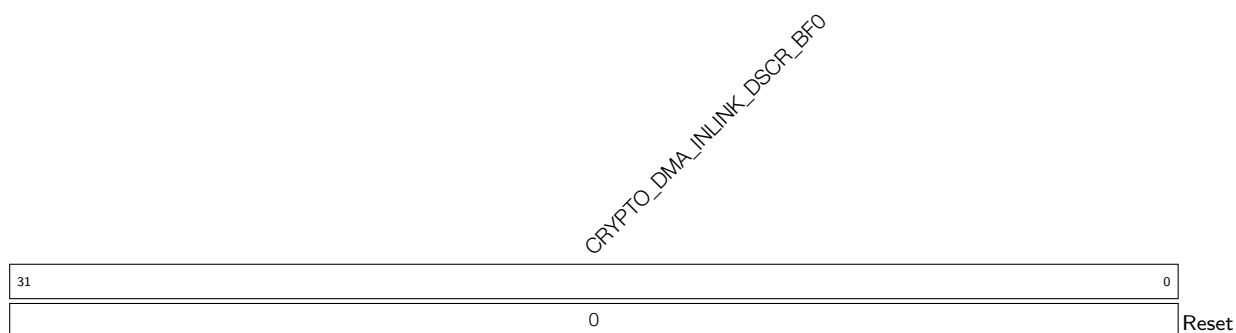
**CRYPTO\_DMA\_IN\_ERR\_EOF\_DES\_ADDR** This register stores the address of the receive descriptor when there are some errors in this descriptor. (RO)

**Register 7.37: CRYPTO\_DMA\_OUT\_EOF\_BFR\_DES\_ADDR\_REG (0x0044)**

**CRYPTO\_DMA\_OUT\_EOF\_BFR\_DES\_ADDR** This register stores the address of the transmit descriptor before the last transmit descriptor. (RO)

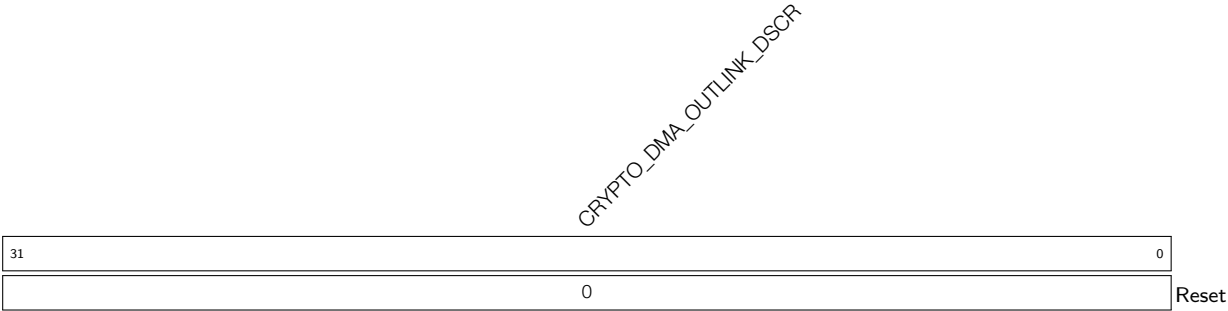
**Register 7.38: CRYPTO\_DMA\_IN\_DSCR\_REG (0x004C)**

**CRYPTO\_DMA\_INLINK\_DSCR** The address of the current receive descriptor x. (RO)

**Register 7.39: CRYPTO\_DMA\_IN\_DSCR\_BF0\_REG (0x0050)**

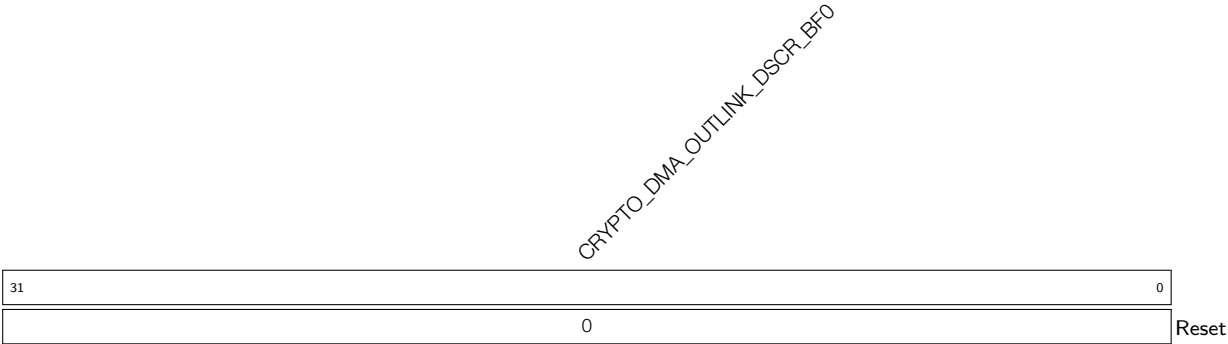
**CRYPTO\_DMA\_INLINK\_DSCR\_BF0** The address of the last receive descriptor x-1. (RO)

Register 7.40: CRYPTO\_DMA\_OUT\_DSCR\_REG (0x0058)



**CRYPTO\_DMA\_OUTLINK\_DSCR** The address of the current transmit descriptor y. (RO)

Register 7.41: CRYPTO\_DMA\_OUT\_DSCR\_BF0\_REG (0x005C)



**CRYPTO\_DMA\_OUTLINK\_DSCR\_BF0** The address of the last transmit descriptor y-1. (RO)

## 8. UART Controller

### 8.1 Overview

In embedded system applications, data is required to be transferred in a simple way with minimal system resources. This can be achieved by a Universal Asynchronous Receiver/Transmitter (UART), which flexibly exchanges data with other peripheral devices in full duplex mode. ESP32-S2 has two UART controllers compatible with various UART devices. They support Infrared Data Association (IrDA) and RS-485 transmission.

ESP32-S2 has two UART controllers. Each has a group of registers that function identically. In this chapter, the two UART controllers are referred to as UART $n$ , in which  $n$  denotes 0 or 1.

### 8.2 Features

Each UART controller has the following features:

- Programmable baud rate
- 512 x 8-bit RAM shared by TX FIFOs and RX FIFOs of two UART controllers
- Full duplex asynchronous communication
- Automatic baud rate detection
- Data bits ranging from 5 to 8
- Stop bits whose length can be 1, 1.5, 2 or 3 bits
- Parity bits
- Special character AT\_CMD detection
- RS-485 protocol
- IrDA protocol
- High-speed data communication using DMA
- UART as wake-up source
- Software and hardware flow control

### 8.3 Functional Description

#### 8.3.1 UART Introduction

A UART is a character-oriented data link for asynchronous communication between devices. Such communication does not add clock signals to data sent. Therefore, in order to communicate successfully, the transmitter and the receiver must operate at the same baud rate with the same stop bit and parity bit.

A UART data packet usually begins with one start bit, followed by data bits, one parity bit (optional) and one or more stop bits. UART controllers on ESP32-S2 support various lengths of data bits and stop bits. These controllers also support software and hardware flow control as well as DMA for seamless high-speed data transfer. This allows developers to use multiple UART ports at minimal software cost.

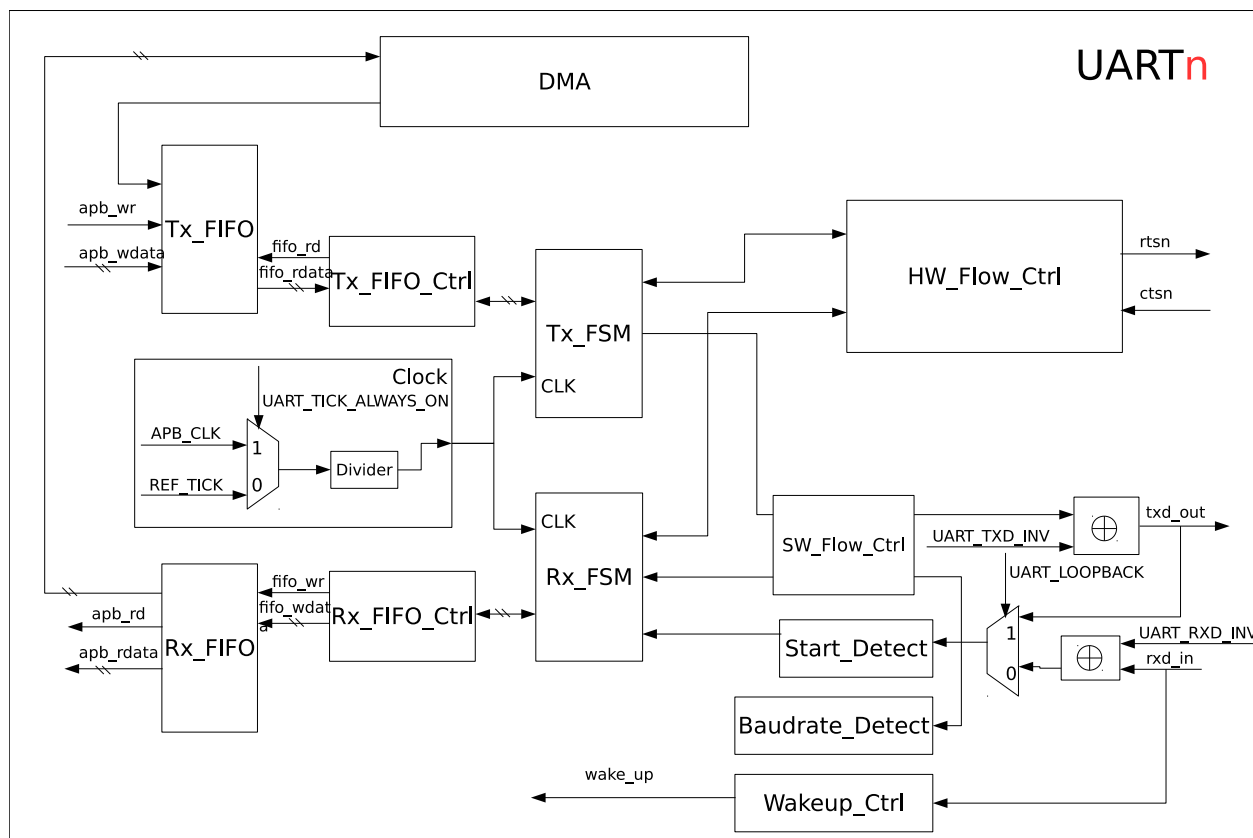


Figure 8-1. UART Structure

### 8.3.2 UART Structure

Figure 8-1 shows the basic structure of a UART controller. It has two possible clock sources: a 80 MHz APB\_CLK and a reference clock REF\_TICK (for details, please refer to Chapter 2 *Reset and Clock*), which are selected by configuring `UART_TICK_REF_ALWAYS_ON`. The selected clock source is divided by a divider to generate clock signals that drive the UART controller. The divisor is configured by `UART_CLKDIV_REG`: `UART_CLKDIV` for the integral part, and `UART_CLKDIV_FRAG` for the fractional part.

A UART controller is broken down into two parts according to functions: a transmitter and a receiver.

The transmitter contains a TX FIFO, which buffers data to be sent. Software can write data to Tx\_FIFO via the APB bus, or move data to Tx\_FIFO using DMA. Tx\_FIFO\_Ctrl controls writing and reading Tx\_FIFO. When Tx\_FIFO is not empty, Tx\_FSM reads bytes via Tx\_FIFO\_Ctrl, and converts them into a bitstream. The levels of output signal txd\_out can be inverted by configuring `UART_TXD_INV` register.

The receiver contains a RX FIFO, which buffers data to be processed. Software can read data from Rx\_FIFO via the APB bus, or receive data using DMA. The levels of input signal rxd\_in can be inverted by configuring `UART_RXD_INV` register, and the signal is then input to the Rx components of the UART Controller:

Baudrate\_Detect measures the baud rate of input signal rxd\_in by detecting its minimum pulse width.

Start\_Detect detects the start bit in a data frame. If the start bit is detected, Rx\_FSM stores data bits in the data frame into Rx\_FIFO by Rx\_FIFO\_Ctrl.

HW\_Flow\_Ctrl controls rxd\_in and txd\_out data flows by standard UART RTS and CTS flow control signals (rtsn\_out and ctsn\_in). SW\_Flow\_Ctrl controls data flows by automatically adding special characters to outgoing data and detecting special characters in incoming data. When a UART controller is in the Light-sleep mode, Wakeup\_Ctrl counts up rising edges of rxd\_in. When the number reaches (`UART_ACTIVE_THRESHOLD` + 2), a



wake\_up signal is generated and sent to RTC, which then wakes up the ESP32-S2 chip.

### 8.3.3 UART RAM

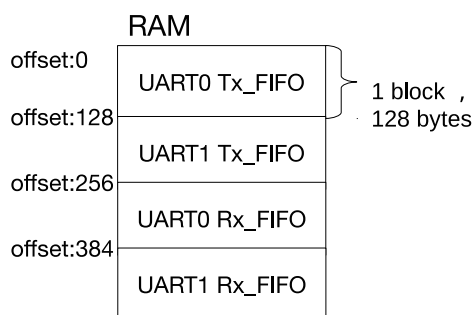


Figure 8-2. UART Controllers Sharing RAM

The two UART controllers on ESP32-S2 share  $512 \times 8$  bits of FIFO RAM. As figure 8-2 illustrates, RAM is divided into 4 blocks, each has  $128 \times 8$  bits. Figure 8-2 shows by default how many RAM blocks are allocated to TX FIFOs and RX FIFOs of the two UART controllers. UART $n$  Tx\_FIFO can be expanded by configuring [UART\\_TX\\_SIZE](#), while UART $n$  Rx\_FIFO can be expanded by configuring [UART\\_RX\\_SIZE](#). The size of UART0 Tx\_FIFO can be increased to 4 blocks (the whole RAM), the size of UART1 Tx\_FIFO can be increased to 3 blocks (from offset 128 to the end address), the size of UART0 Rx\_FIFO can be increased to 2 blocks (from offset 256 to the end address), but the size of UART1 Rx\_FIFO cannot be increased. Please note that expanding one FIFO may take up the default space of other FIFOs. For example, by setting [UART\\_TX\\_SIZE](#) of UART0 to 2, the size of UART0 Tx\_FIFO is increased by 128 bytes (from offset 0 to offset 255). In this case, UART0 Tx\_FIFO takes up the default space for UART1 Tx\_FIFO, and UART1's transmitting function cannot be used as a result.

When neither of the two UART controllers is active, RAM could enter low-power mode by setting [UART\\_MEM\\_FORCE\\_PD](#).

UART0 Tx\_FIFO and UART1 Tx\_FIFO are reset by setting [UART\\_TXFIFO\\_RST](#). UART0 Rx\_FIFO and UART1 Rx\_FIFO are reset by setting [UART\\_RXFIFO\\_RST](#).

Data to be sent is written to TX FIFO via the APB bus or using DMA, read automatically and converted from a frame into a bitstream by hardware Tx\_FSM; data received is converted from a bitstream into a frame by hardware Rx\_FSM, written into RX FIFO, and then stored into RAM via the APB bus or using DMA. The two UART controllers share one DMA controller.

The empty signal threshold for Tx\_FIFO is configured by setting [UART\\_TXFIFO\\_EMPTY\\_THRHD](#). When data stored in Tx\_FIFO is less than [UART\\_TXFIFO\\_EMPTY\\_THRHD](#), a UART\_TXFIFO\_EMPTY\_INT interrupt is generated.

The full signal threshold for Rx\_FIFO is configured by setting [UART\\_RXFIFO\\_FULL\\_THRHD](#). When data stored in Rx\_FIFO is greater than [UART\\_RXFIFO\\_FULL\\_THRHD](#), a UART\_RXFIFO\_FULL\_INT interrupt is generated. In addition, when Rx\_FIFO receives more data than its capacity, a UART\_RXFIFO\_OVF\_INT interrupt is generated.

UART $n$  can access FIFO via register UART\_FIFO\_REG.

### 8.3.4 Baud Rate Generation and Detection

### 8.3.4.1 Baud Rate Generation

Before a UART controller sends or receives data, the baud rate should be configured by setting corresponding registers. A UART Controller baud rate generator functions by dividing the input clock source. It can divide the clock source by a fractional amount. The divisor is configured by `UART_CLKDIV_REG: UART_CLKDIV` for the integral part, and `UART_CLKDIV_FRAG` for the fractional part. When using an 80 MHz input clock, the UART controller supports a maximum baud rate of 5 Mbaud.

The divisor of the baud rate divider is equal to  $\text{UART\_CLKDIV} + (\text{UART\_CLKDIV\_FRAG} / 16)$ , meaning that the final baud rate is equal to  $\text{INPUT\_FREQ} / (\text{UART\_CLKDIV} + (\text{UART\_CLKDIV\_FRAG} / 16))$ . For example, if  $\text{UART\_CLKDIV} = 694$  and  $\text{UART\_CLKDIV\_FRAG} = 7$  then the divisor value is  $(694 + 7/16) = 694.4375$ . If the input clock frequency is 80MHz APB\_CLK, the baud rate will be  $(80\text{MHz} / 69.4375) = 115201$ .

When `UART_CLKDIV_FRAG` is zero, the baud rate generator is an integer clock divider where an output pulse is generated every `UART_CLKDIV` input pulses.

When `UART_CLKDIV_FRAG` is not zero, the divider is fractional and the output baud rate clock pulses are not strictly uniform. As shown in figure 8-3, for every 16 output pulses, the generator divides either  $(\text{UART\_CLKDIV} + 1)$  input pulses or `UART_CLKDIV` input pulses per output pulse. A total of `UART_CLKDIV_FRAG` output pulses are generated by dividing  $(\text{UART\_CLKDIV} + 1)$  input pulses, and the remaining  $(16 - \text{UART\_CLKDIV\_FRAG})$  output pulses are generated by dividing `UART_CLKDIV` input pulses.

The output pulses are interleaved as shown in figure 8-3 below, to make the output timing more uniform:

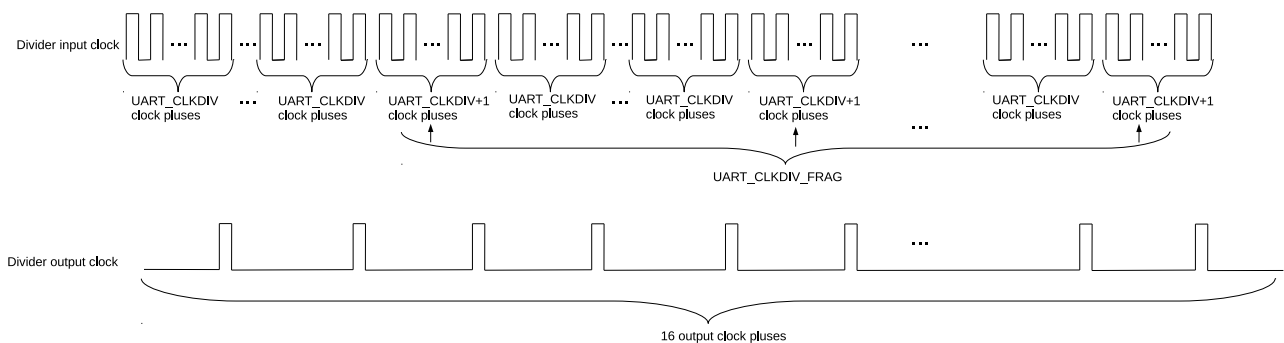


Figure 8-3. UART Controllers Division

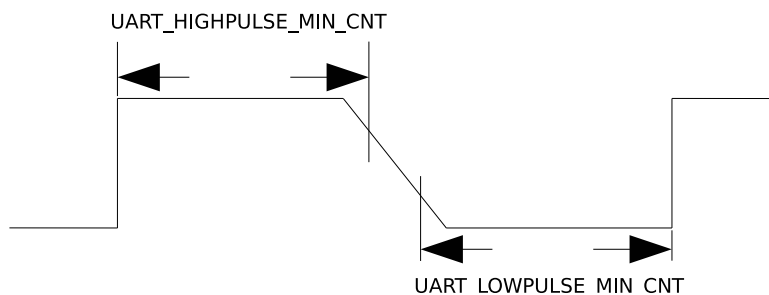
To support IrDA (see Section 8.3.7 *IrDA* for details), the fractional clock divider for IrDA data transmission generates clock signals divided by  $16 \times \text{UART\_CLKDIV\_REG}$ . This divider works similarly as the one elaborated above: it takes  $\text{UART\_CLKDIV}/16$  as the integer value and the lowest four bits of `UART_CLKDIV` as the fractional value.

### 8.3.4.2 Baud Rate Detection

Automatic baud rate detection (Autobaud) on UARTs is enabled by setting `UART_AUTOBAUD_EN`. The Baudrate\_Detect module shown in figure 8-1 will measure pulse widths while filtering any noise whose pulse width is shorter than `UART_GLITCH_FILT`.

Before communication starts, the transmitter could send random data to the receiver for baud rate detection. `UART_LOWPULSE_MIN_CNT` stores the minimum low pulse width, `UART_HIGHPULSE_MIN_CNT` stores the minimum high pulse width, `UART_POSEDGE_MIN_CNT` stores the minimum pulse width between two rising edges, and `UART_NEGEDGE_MIN_CNT` stores the minimum pulse width between two falling edges. These four

registers are read by software to determine the transmitter's baud rate.



**Figure 8-4. The Timing Diagram of Weak UART Signals Along Falling Edges**

Baud rate can be determined in the following three ways:

1. Normally, to avoid sampling erroneous data along rising or falling edges in semi-stable state, which results in inaccuracy of `UART_LOWPULSE_MIN_CNT` or `UART_HIGHPULSE_MIN_CNT`, use a weighted average of these two values to eliminate errors. In this case, baud rate is calculated as follows:

$$B_{\text{uart}} = \frac{f_{\text{clk}}}{(\text{UART\_LOWPULSE\_MIN\_CNT} + \text{UART\_HIGHPULSE\_MIN\_CNT})/2}$$

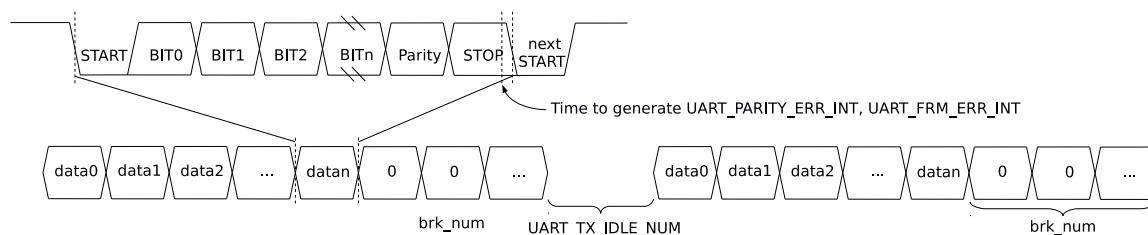
2. If UART signals are weak along falling edges as shown in figure 8-4, which leads to inaccurate average of `UART_LOWPULSE_MIN_CNT` and `UART_HIGHPULSE_MIN_CNT`, use `UART_POSEDGE_MIN_CNT` to determine the transmitter's baud rate as follows:

$$B_{\text{uart}} = \frac{f_{\text{clk}}}{\text{UART\_POSEDGE\_MIN\_CNT}/2}$$

3. If UART signals are weak along rising edges, use `UART_NEGEDGE_MIN_CNT` to determine the transmitter's baud rate as follows:

$$B_{\text{uart}} = \frac{f_{\text{clk}}}{\text{UART\_NEGEDGE\_MIN\_CNT}/2}$$

### 8.3.5 UART Data Frame

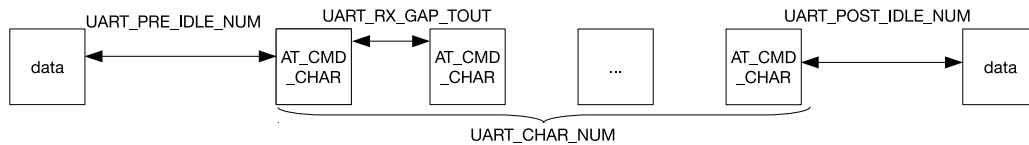


**Figure 8-5. Structure of UART Data Frame**

Figure 8-5 shows the basic structure of a data frame. A frame starts with one START bit, and ends with STOP bits which can be 1, 1.5, 2 or 3 bits long, configured by `UART_STOP_BIT_NUM`, `UART_DL1_EN` and `UART_DLO_EN`. The START bit is logical low, whereas STOP bits are logical high.

The actual data length can be anywhere between 5 ~ 8 bit, configured by `UART_BIT_NUM`. When `UART_PARITY_EN` is set, a parity bit is added after data bits. `UART_PARITY` is used to choose even parity or odd parity. When the receiver detects a parity bit error in data received, a `UART_PARITY_ERR_INT` interrupt is generated and data received is still stored into RX FIFO. When the receiver detects a data frame error, a `UART_FRM_ERR_INT` interrupt is generated, and data received by default is stored into RX FIFO.

If all data in Tx\_FIFO has been sent, a UART\_TX\_DONE\_INT interrupt is generated. After this, if the [UART\\_TXD\\_BRK](#) bit is set then the transmitter will send several NULL characters in which the TX data line is logical low. The number of NULL characters is configured by [UART\\_TX\\_BRK\\_NUM](#). Once the transmitter has sent all NULL characters, a UART\_TX\_BRK\_DONE\_INT interrupt is generated. The minimum interval between data frames can be configured using [UART\\_TX\\_IDLE\\_NUM](#). If the transmitter stays idle for [UART\\_TX\\_IDLE\\_NUM](#) or more time, a UART\_TX\_BRK\_IDLE\_DONE\_INT interrupt is generated.



**Figure 8-6. AT\_CMD Character Structure**

Figure 8-6 is the structure of a special character AT\_CMD. If the receiver constantly receives AT\_CMD\_CHAR and the following conditions are met, a UART\_AT\_CMD\_CHAR\_DET\_INT interrupt is generated.

- The interval between the first AT\_CMD\_CHAR and the last non AT\_CMD\_CHAR character is at least [UART\\_PRE\\_IDLE\\_NUM](#) single-bit cycles.
- The interval between two AT\_CMD\_CHAR characters is less than [UART\\_RX\\_GAP\\_TOUT](#) single-bit cycles.
- The number of AT\_CMD\_CHAR characters is equal to or greater than [UART\\_CHAR\\_NUM](#).
- The interval between the last AT\_CMD\_CHAR character and next non AT\_CMD\_CHAR character is at least [UART\\_POST\\_IDLE\\_NUM](#) single-bit cycles.

### 8.3.6 RS485

The two UART controllers support RS485 standard. This standard uses differential signals to transmit data, so it can communicate over longer distances at higher bit rates than RS232. RS485 has two-wire half-duplex mode and four-wire full-duplex mode. UART controllers support two-wire half-duplex transmission and bus snooping. In a two-wire RS485 multidrop network, there can be 32 slaves at most.

#### 8.3.6.1 Driver Control

As shown in figure 8-7, in a two-wire multidrop network, an external RS485 transceiver is needed for differential to single-ended conversion. A RS485 transceiver contains a driver and a receiver. When a UART controller is not in transmitter mode, the connection to the differential line can be broken by disabling the driver. When DE is 1, the driver is enabled; when DE is 0, the driver is disabled.

The receiving UART converts differential signals to single-ended signals via an external receiver. RE is the enable control signal for the receiver. When RE is 0, the receiver is enabled; when RE is 1, the receiver is disabled. If RE is configured as 0, the UART controller is allowed to snoop data on the bus, including data sent by itself.

DE can be controlled by either software or hardware. To reduce cost of software, in our design DE is controlled by hardware. As shown in figure 8-7, DE is connected to dtrn\_out of UART (please refer to Section [8.3.9.1 Hardware Flow Control](#) for more details).

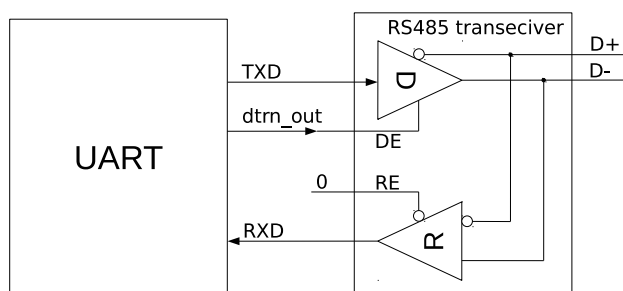


Figure 8-7. Driver Control Diagram in RS485 Mode

### 8.3.6.2 Turnaround Delay

By default, the two UART controllers work in receiver mode. When a UART controller is switched from transmitter mode to receiver mode, the RS485 protocol requires a turnaround delay of at least one cycle after the stop bit. The transmitter supports turnaround delay of two cycles added after the stop bit. When `UART_DL1_EN` is set, turnaround delay of one single-bit cycle is added; when `UART_DL0_EN` is set, turnaround delay of a second cycle is added.

### 8.3.6.3 Bus Snooping

By default, an RS485 device is not allowed to transmit and receive data simultaneously. However, the UART controller peripheral supports snooping this bus by receiving while transmitting. If `UART_RS485TX_RX_EN` is set and the external RS485 transceiver is configured as in figure 8-7, a UART controller may receive data in transmitter mode and snoop the bus. If `UART_RS485RXBY_TX_EN` is set, a UART controller may transmit data in receiver mode.

The two UART controllers can snoop data sent by themselves. In transmitter mode, when a UART controller monitors a collision between data sent and data received, a `UART_RS485_CLASH_INT` is generated; when a UART controller monitor a data frame error, a `UART_RS485_FRM_ERR_INT` interrupt is generated; when a UART controller monitors a polarity error, a `UART_RS485_PARITY_ERR_INT` is generated.

### 8.3.7 IrDA

IrDA protocol consists of three layers, namely the physical layer, the link access protocol and the link management protocol. The two UART controllers implement IrDA physical layer. In IrDA encoding, a UART controller supports data rates up to 115.2 kbit/s (SIR, or serial infrared mode). As shown in figure 8-8, the IrDA encoder converts a NRZ (non-return to zero code) signal to a RZI (return to zero code) signal and sends it to the external driver and infrared LED. This encoder uses modulated signals whose pulse width is 3/16 bits to indicate logic “0”, and low levels to indicate logic “1”. The IrDA decoder receives signals from the infrared receiver and converts them to NRZ signals. In most cases, the receiver is high when it is idle, and the encoder output polarity is the opposite of the decoder input polarity. If a low pulse is detected, it indicates that a start bit has been received.

When IrDA function is enabled, one bit is divided into 16 clock cycles. If the bit to be sent is zero, then the 9th, 10th and 11th clock cycle is high.

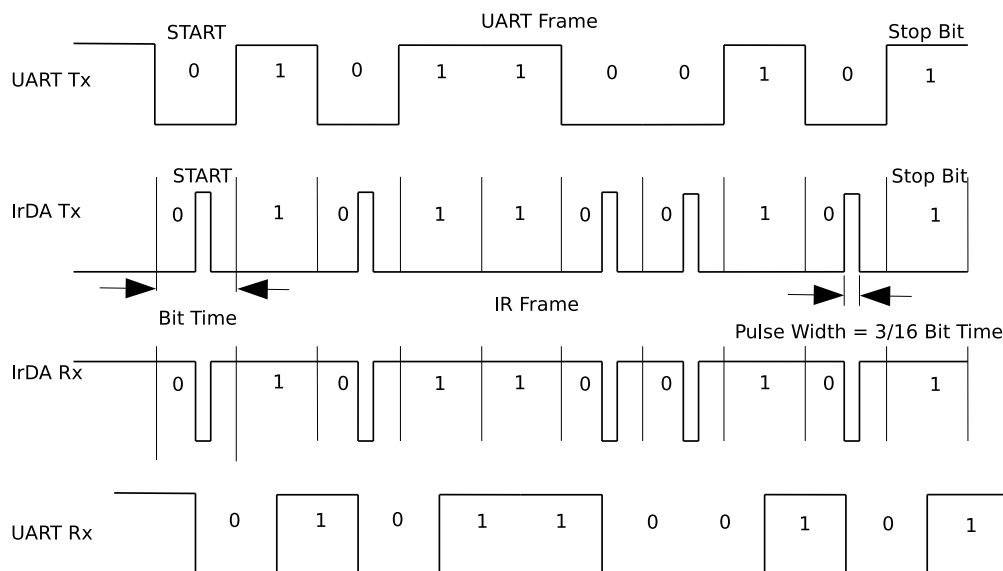


Figure 8-8. The Timing Diagram of Encoding and Decoding in SIR mode

The IrDA transceiver is half-duplex, meaning that it cannot send and receive data simultaneously. As shown in figure 8-9, IrDA function is enabled by setting `UART_IRDA_EN`. When `UART_IRDA_TX_EN` is set (high), the IrDA transceiver is enabled to send data and not allowed to receive data; when `UART_IRDA_TX_EN` is reset (low), the IrDA transceiver is enabled to receive data and not allowed to send data.

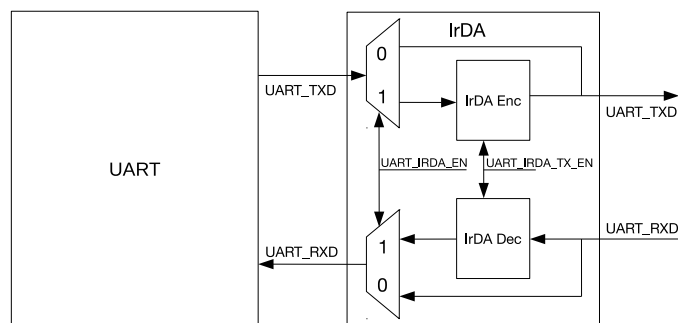


Figure 8-9. IrDA Encoding and Decoding Diagram

### 8.3.8 Wake-up

UART0 and UART1 can be set as wake-up source. When a UART controller is in Light-sleep mode, Wakeup\_Ctrl counts up the rising edges of rxd\_in. When the number of rising edges is greater than (`UART_ACTIVE_THRESHOLD` + 2), a wake\_up signal is generated and sent to RTC, which then wakes up ESP32-S2.

### 8.3.9 Flow Control

UART controllers have two ways to control data flow, namely hardware flow control and software flow control. Hardware flow control is achieved using output signal rtsn\_out and input signal dsrn\_in. Software flow control is achieved by inserting special characters in data flow sent and detecting special characters in data flow received.

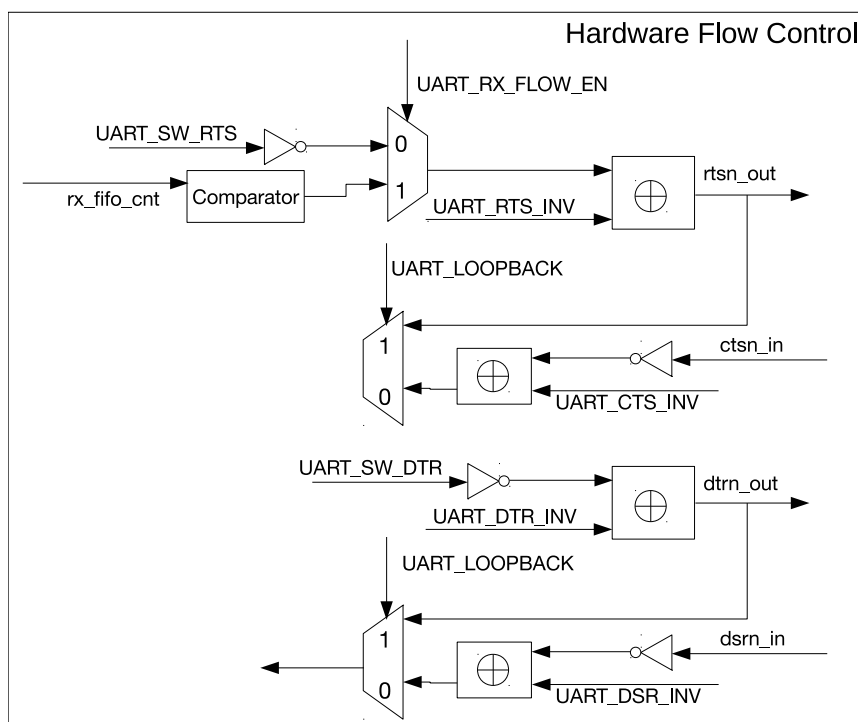


Figure 8-10. Hardware Flow Control Diagram

### 8.3.9.1 Hardware Flow Control

Figure 8-10 shows hardware flow control of a UART controller. Hardware flow control uses output signal `rtsn_out` and input signal `dsrn_in`. Figure 8-11 illustrates how these signals are connected between ESP32-S2 UART (hereinafter referred to as IU0) and the external UART (hereinafter referred to as EU0).

When `rtsn_out` of IU0 is low, EU0 is allowed to send data; when `rtsn_out` of IU0 is high, EU0 is notified to stop sending data until `rtsn_out` of IU0 returns to low. Output signal `rtsn_out` can be controlled in two ways.

- Software control: Enter this mode by setting `UART_RX_FLOW_EN` to 0. In this mode, the level of `rtsn_out` is changed by configuring `UART_SW_RTS`.
- Hardware control: Enter this mode by setting `UART_RX_FLOW_EN` to 1. In this mode, `rtsn_out` is pulled high when data in Rx\_FIFO exceeds `UART_RX_FLOW_THRHD`.

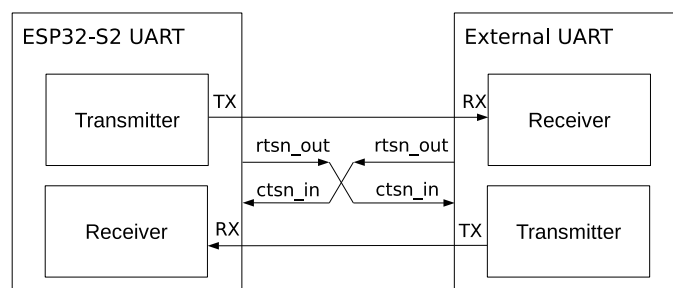


Figure 8-11. Connection between Hardware Flow Control Signals

When `cts_in` of IU0 is low, IU0 is allowed to send data; when `cts_in` is high, IU0 is not allowed to send data. When IU0 detects an edge change of `cts_in`, a `UART_CTS_CHG_INT` interrupt is generated.

If `dtrn_out` of IU0 is high, it indicates that IU0 is ready to transmit data. `dtrn_out` is generated by configuring register `UART_SW_DTR`. When the IU0 transmitter detects a edge change of `dsrn_in`, a `UART_DSR_CHG_INT`

interrupt is generated. After this interrupt is detected, software can obtain the level of input signal `dsrn_in` by reading `UART_DSRN`. If `dsrn_in` is high, it indicates that EU0 is ready to transmit data.

In a two-wire RS485 multidrop network enabled by setting `UART_RS485_EN`, `dtrn_out` is used for transmit/receive turnaround. In this case, `dtrn_out` is generated by hardware. When data transmission starts, `dtrn_out` is pulled high and the external driver is enabled; when data transmission completes, `dtrn_out` is pulled low and the external driver is disabled. Please note that when there is turnaround delay of one cycle added after the stop bit, `dtrn_out` is pulled low after the delay.

UART loopback test is enabled by setting `UART_LOOPBACK`. In the test, UART output signal `txd_out` is connected to its input signal `rx_d_in`, `rtsn_out` is connected to `ctsn_in`, and `dtrn_out` is connected to `dsrn_out`. If data sent matches data received, it indicates that UART controllers are working properly.

### 8.3.9.2 Software Flow Control

Instead of CTS/RTS lines, software flow control uses XON/XOFF characters to start or stop data transmission. Such flow control is enabled by setting `UART_SW_FLOW_CON_EN` to 1.

When using software flow control, hardware automatically detects if there are XON/XOFF characters in data flow received, and generate a `UART_SW_XOFF_INT` or a `UART_SW_XON_INT` interrupt accordingly. If an XOFF character is detected, the transmitter stops data transmission once the current byte has been transmitted; if an XON character is detected, the transmitter starts data transmission. In addition, software can force the transmitter to stop sending data by setting `UART_FORCE_XOFF`, or to start sending data by setting `UART_FORCE_XON`.

Software determines whether to insert flow control characters according to the remaining room in RX FIFO. When `UART_SEND_XOFF` is set, the transmitter sends an XOFF character configured by `UART_XOFF_CHAR` after the current byte in transmission; when `UART_SEND_XON` is set, the transmitter sends an XON character configured by `UART_XON_CHAR` after the current byte in transmission. If the RX FIFO of a UART controller stores more data than `UART_XOFF_THRESHOLD`, `UART_SEND_XOFF` is set by hardware. As a result, the transmitter sends an XOFF character after the current byte in transmission. If the RX FIFO of a UART controller stores less data than `UART_XON_THRESHOLD`, `UART_SEND_XON` is set by hardware. As a result, the transmitter sends an XON character after the current byte in transmission.

### 8.3.10 UDMA

The two UART controllers on ESP32-S2 share one UDMA (UART DMA), which supports the decoding and encoding of HCI data packets. For more information, please refer to Chapter 7: [DMA Controller](#).

### 8.3.11 UART Interrupts

- `UART_AT_CMD_CHAR_DET_INT`: Triggered when the receiver detects an `AT_CMD` character.
- `UART_RS485_CLASH_INT`: Triggered when a collision is detected between the transmitter and the receiver in RS485 mode.
- `UART_RS485_FRM_ERR_INT`: Triggered when an error is detected in the data frame sent by the transmitter in RS485 mode.
- `UART_RS485_PARITY_ERR_INT`: Triggered when an error is detected in the parity bit sent by the transmitter in RS485 mode.
- `UART_TX_DONE_INT`: Triggered when all data in the transmitter's TX FIFO has been sent.



- `UART_TX_BRK_IDLE_DONE_INT`: Triggered when the transmitter stays idle for the minimum interval (threshold) after sending the last data bit.
- `UART_TX_BRK_DONE_INT`: Triggered when the transmitter sends a NULL character after all data in TX FIFO has been sent.
- `UART_GLITCH_DET_INT`: Triggered when the receiver detects a glitch in the middle of the start bit.
- `UART_SW_XOFF_INT`: Triggered when `UART_SW_FLOW_CON_EN` is set and the receiver receives a Xoff character.
- `UART_SW_XON_INT`: Triggered when `UART_SW_FLOW_CON_EN` is set and the receiver receives a Xon character.
- `UART_RXFIFO_TOUT_INT`: Triggered when the receiver takes more time than `UART_RX_TOUT_THRHD` to receive one byte.
- `UART_BRK_DET_INT`: Triggered when the receiver detects a NULL character after stop bits.
- `UART_CTS_CHG_INT`: Triggered when the receiver detects an edge change of CTSn signals.
- `UART_DSR_CHG_INT`: Triggered when the receiver detects an edge change of DSRn signals.
- `UART_RXFIFO_OVF_INT`: Triggered when the receiver receives more data than the capacity of RX FIFO.
- `UART_FRM_ERR_INT`: Triggered when the receiver detects a data frame error.
- `UART_PARITY_ERR_INT`: Triggered when the receiver detects a parity error.
- `UART_TXFIFO_EMPTY_INT`: Triggered when TX FIFO stores less data than what `UART_TXFIFO_EMPTY_THRHD` specifies.
- `UART_RXFIFO_FULL_INT`: Triggered when the receiver receives more data than what `UART_RXFIFO_FULL_THRHD` specifies.
- `UART_WAKEUP_INT`: Triggered when UART is woken up.

### 8.3.12 UHCI Interrupts

- `UHCI_DMA_INFIFO_FULL_WM_INT`: Triggered when the counter value of DMA RX FIFO exceeds `UHCI_DMA_INFIFO_FULL_THRS`.
- `UHCI_SEND_A_REG_Q_INT`: Triggered when DMA has sent a series of short packets using `always_send`.
- `UHCI_SEND_S_REG_Q_INT`: Triggered when DMA has sent a series of short packets using `single_send`.
- `UHCI_OUT_TOTAL_EOF_INT`: Triggered when all data has been sent.
- `UHCI_OUTLINK_EOF_ERR_INT`: Triggered when an EOF error is detected in a transmit descriptor.
- `UHCI_IN_DSCR_EMPTY_INT`: Triggered when there are not enough receive descriptors for DMA.
- `UHCI_OUT_DSCR_ERR_INT`: Triggered when an error is detected in a transmit descriptor.
- `UHCI_IN_DSCR_ERR_INT`: Triggered when an error is detected in an receive descriptor.
- `UHCI_OUT_EOF_INT`: Triggered when the EOF bit in a descriptor is 1.
- `UHCI_OUT_DONE_INT`: Triggered when a transmit descriptor is completed.
- `UHCI_IN_ERR_EOF_INT`: Triggered when an EOF error is detected in an receive descriptor.

- UHCI\_IN\_SUC\_EOF\_INT: Triggered when a data packet has been received.
- UHCI\_IN\_DONE\_INT: Triggered when an receive descriptor is completed.
- UHCI\_TX\_HUNG\_INT: Triggered when DMA spends too much time on reading RAM.
- UHCI\_RX\_HUNG\_INT: Triggered when DMA spends too much time on receiving data.
- UHCI\_TX\_START\_INT: Triggered when DMA detects a separator character.
- UHCI\_RX\_START\_INT: Triggered when a separator character has been sent.

## 8.4 Base Address

Users can access UART0, UART1 and UHCI0 respectively with two register base addresses shown in the following table. For more information about accessing peripherals from different buses please see Chapter 1 [System and Memory](#).

**Table 40: UART0, UART1 and UHCI0 Base Address**

Module	Bus to Access Peripheral	Base Address
UART0	PeriBUS1	0x3F400000
	PeriBUS2	0x60000000
UART1	PeriBUS1	0x3F410000
	PeriBUS2	0x60010000
UHCI0	PeriBUS1	0x3F414000
	PeriBUS2	0x60014000

## 8.5 Register Summary

The addresses in the following table are relative to the UART base addresses provided in Section 8.4.

Name	Description	Address	Access
<b>FIFO Configuration</b>			
<a href="#">UART_FIFO_REG</a>	FIFO data register	0x0000	R/W
<a href="#">UART_MEM_CONF_REG</a>	UART threshold and allocation configuration	0x005C	R/W
<b>Interrupt registers</b>			
<a href="#">UART_INT_RAW_REG</a>	Raw interrupt status	0x0004	RO
<b>Interrupt Register</b>			
<a href="#">UART_INT_ST_REG</a>	Masked interrupt status	0x0008	RO
<a href="#">UART_INT_ENA_REG</a>	Interrupt enable bits	0x000C	R/W
<a href="#">UART_INT_CLR_REG</a>	Interrupt clear bits	0x0010	WO
<b>Configuration Register</b>			
<a href="#">UART_CLKDIV_REG</a>	Clock divider configuration	0x0014	R/W
<a href="#">UART_CONF0_REG</a>	Configuration register 0	0x0020	R/W
<a href="#">UART_CONF1_REG</a>	Configuration register 1	0x0024	R/W
<a href="#">UART_FLOW_CONF_REG</a>	Software flow control configuration	0x0034	R/W
<a href="#">UART_SLEEP_CONF_REG</a>	Sleeping mode configuration	0x0038	R/W
<a href="#">UART_SWFC_CONF0_REG</a>	Software flow control character configuration	0x003C	R/W
<a href="#">UART_SWFC_CONF1_REG</a>	Software flow-control character configuration	0x0040	R/W

Name	Description	Address	Access
<a href="#">UART_IDLE_CONF_REG</a>	Frame-end idle configuration	0x0044	R/W
<a href="#">UART_RS485_CONF_REG</a>	RS485 mode configuration	0x0048	R/W
<b>Autobaud Register</b>			
<a href="#">UART_AUTOBAUD_REG</a>	Autobaud configuration register	0x0018	R/W
<a href="#">UART_LOWPULSE_REG</a>	Autobaud minimum low pulse duration register	0x0028	RO
<a href="#">UART_HIGHPULSE_REG</a>	Autobaud minimum high pulse duration register	0x002C	RO
<a href="#">UART_RXD_CNT_REG</a>	Autobaud edge change count register	0x0030	RO
<a href="#">UART_POSPULSE_REG</a>	Autobaud high pulse register	0x006C	RO
<a href="#">UART_NEGPULSE_REG</a>	Autobaud low pulse register	0x0070	RO
<b>Status Register</b>			
<a href="#">UART_STATUS_REG</a>	UART status register	0x001C	RO
<a href="#">UART_MEM_TX_STATUS_REG</a>	TX FIFO write and read offset address	0x0060	RO
<a href="#">UART_MEM_RX_STATUS_REG</a>	RX FIFO write and read offset address	0x0064	RO
<a href="#">UART_FSM_STATUS_REG</a>	UART transmit and receive status	0x0068	RO
<b>AT Escape Sequence Selection Configuration</b>			
<a href="#">UART_AT_CMD_PRECNT_REG</a>	Pre-sequence timing configuration	0x004C	R/W
<a href="#">UART_AT_CMD_POSTCNT_REG</a>	Post-sequence timing configuration	0x0050	R/W
<a href="#">UART_AT_CMD_GAPTOOUT_REG</a>	Timeout configuration	0x0054	R/W
<a href="#">UART_AT_CMD_CHAR_REG</a>	AT Escape Sequence Selection Configuration	0x0058	R/W
<b>Version Register</b>			
<a href="#">UART_DATE_REG</a>	UART version control register	0x0074	R/W

Name	Description	Address	Access
<b>Configuration Register</b>			
<a href="#">UHCI_CONF0_REG</a>	UHCI configuration register	0x0000	R/W
<a href="#">UHCI_CONF1_REG</a>	UHCI configuration register	0x002C	R/W
<a href="#">UHCI_AHB_TEST_REG</a>	AHB test register	0x0048	R/W
<a href="#">UHCI_ESCAPE_CONF_REG</a>	Escape characters configuration	0x0064	R/W
<a href="#">UHCI_HUNG_CONF_REG</a>	Timeout configuration	0x0068	R/W
<a href="#">UHCI_QUICK_SENT_REG</a>	UHCI quick send configuration register	0x0074	R/W
<a href="#">UHCI_Q0_WORD0_REG</a>	Q0_WORD0 quick_sent register	0x0078	R/W
<a href="#">UHCI_Q0_WORD1_REG</a>	Q0_WORD1 quick_sent register	0x007C	R/W
<a href="#">UHCI_Q1_WORD0_REG</a>	Q1_WORD0 quick_sent register	0x0080	R/W
<a href="#">UHCI_Q1_WORD1_REG</a>	Q1_WORD1 quick_sent register	0x0084	R/W
<a href="#">UHCI_Q2_WORD0_REG</a>	Q2_WORD0 quick_sent register	0x0088	R/W
<a href="#">UHCI_Q2_WORD1_REG</a>	Q2_WORD1 quick_sent register	0x008C	R/W
<a href="#">UHCI_Q3_WORD0_REG</a>	Q3_WORD0 quick_sent register	0x0090	R/W
<a href="#">UHCI_Q3_WORD1_REG</a>	Q3_WORD1 quick_sent register	0x0094	R/W
<a href="#">UHCI_Q4_WORD0_REG</a>	Q4_WORD0 quick_sent register	0x0098	R/W
<a href="#">UHCI_Q4_WORD1_REG</a>	Q4_WORD1 quick_sent register	0x009C	R/W
<a href="#">UHCI_Q5_WORD0_REG</a>	Q5_WORD0 quick_sent register	0x00A0	R/W
<a href="#">UHCI_Q5_WORD1_REG</a>	Q5_WORD1 quick_sent register	0x00A4	R/W
<a href="#">UHCI_Q6_WORD0_REG</a>	Q6_WORD0 quick_sent register	0x00A8	R/W

Name	Description	Address	Access
<a href="#">UHCI_Q6_WORD1_REG</a>	Q6_WORD1 quick_sent register	0x00AC	R/W
<a href="#">UHCI_ESC_CONF0_REG</a>	Escape sequence configuration register 0	0x00B0	R/W
<a href="#">UHCI_ESC_CONF1_REG</a>	Escape sequence configuration register 1	0x00B4	R/W
<a href="#">UHCI_ESC_CONF2_REG</a>	Escape sequence configuration register 2	0x00B8	R/W
<a href="#">UHCI_ESC_CONF3_REG</a>	Escape sequence configuration register 3	0x00BC	R/W
<a href="#">UHCI_PKT_THRES_REG</a>	Configure register for packet length	0x00C0	R/W
<b>Interrupt Register</b>			
<a href="#">UHCI_INT_RAW_REG</a>	Raw interrupt status	0x0004	RO
<a href="#">UHCI_INT_ST_REG</a>	Masked interrupt status	0x0008	RO
<a href="#">UHCI_INT_ENA_REG</a>	Interrupt enable bits	0x000C	R/W
<a href="#">UHCI_INT_CLR_REG</a>	Interrupt clear bits	0x0010	WO
<b>DMA Status</b>			
<a href="#">UHCI_DMA_OUT_STATUS_REG</a>	DMA data-output status register	0x0014	RO
<a href="#">UHCI_DMA_IN_STATUS_REG</a>	UHCI data-input status register	0x001C	RO
<a href="#">UHCI_STATE0_REG</a>	UHCI decoder status register	0x0030	RO
<a href="#">UHCI_STATE1_REG</a>	UHCI encoder status register	0x0034	RO
<a href="#">UHCI_DMA_OUT_EOF_DES_ADDR_REG</a>	Outlink descriptor address when EOF occurs	0x0038	RO
<a href="#">UHCI_DMA_IN_SUC_EOF_DES_ADDR_REG</a>	Inlink descriptor address when EOF occurs	0x003C	RO
<a href="#">UHCI_DMA_IN_ERR_EOF_DES_ADDR_REG</a>	Inlink descriptor address when errors occur	0x0040	RO
<a href="#">UHCI_DMA_OUT_EOF_BFR_DES_ADDR_REG</a>	Outlink descriptor address before the last transmit descriptor	0x0044	RO
<a href="#">UHCI_DMA_IN_DSCR_REG</a>	The third word of the next receive descriptor	0x004C	RO
<a href="#">UHCI_DMA_IN_DSCR_BF0_REG</a>	The third word of current receive descriptor	0x0050	RO
<a href="#">UHCI_DMA_OUT_DSCR_REG</a>	The third word of the next transmit descriptor	0x0058	RO
<a href="#">UHCI_DMA_OUT_DSCR_BF0_REG</a>	The third word of current transmit descriptor	0x005C	RO
<a href="#">UHCI_RX_HEAD_REG</a>	UHCI packet header register	0x0070	RO
<b>DMA Configuration</b>			
<a href="#">UHCI_DMA_OUT_PUSH_REG</a>	Push control register of data-output FIFO	0x0018	R/W
<a href="#">UHCI_DMA_IN_POP_REG</a>	Pop control register of data-input FIFO	0x0020	varies
<a href="#">UHCI_DMA_OUT_LINK_REG</a>	Link descriptor address and control	0x0024	varies
<a href="#">UHCI_DMA_IN_LINK_REG</a>	Link descriptor address and control	0x0028	varies
<b>Version Register</b>			
<a href="#">UHCI_DATE_REG</a>	UHCI version control register	0x00FC	R/W

## 8.6 Registers

**Register 8.1: UART\_FIFO\_REG (0x0000)**

(reserved)																								UART_RXFIFO_RD_BYTE															
31																								7								0							
0 0																								0x0								Reset							

**UART\_RXFIFO\_RD\_BYTE** UART *n* accesses FIFO via this register. (R/W)

**Register 8.2: UART\_MEM\_CONF\_REG (0x005C)**

(reserved)				UART_MEM_FORCE_PU UART_MEM_FORCE_PD				UART_RX_TOUT_THRHD								UART_RX_FLOW_THRHD								UART_TX_SIZE				UART_RX_SIZE				(reserved)				
31				28				27	26	25				16				15				7				6	4				3	1				0
0				0				0	0	0xa				0x0				0x0				0x1				1				0				Reset		

**UART\_RX\_SIZE** This register is used to configure the amount of mem allocated for RX FIFO. The default number is 128 bytes. (R/W)

**UART\_TX\_SIZE** This register is used to configure the amount of mem allocated for TX FIFO. The default number is 128 bytes. (R/W)

**UART\_RX\_FLOW\_THRHD** This register is used to configure the maximum amount of data that can be received when hardware flow control works. (R/W)

**UART\_RX\_TOUT\_THRHD** This register is used to configure the threshold time that receiver takes to receive one byte. The UART\_RXFIFO\_TOUT\_INT interrupt will be triggered when the receiver takes more time to receive one byte with UART\_RX\_TOUT\_EN set to 1. (R/W)

**UART\_MEM\_FORCE\_PD** Set this bit to force power down UART memory. (R/W)

**UART\_MEM\_FORCE\_PU** Set this bit to force power up UART memory. (R/W)

**Register 8.3: UART\_INT\_RAW\_REG (0x0004)**

(reserved)																UART_WAKEUP_INT_RAW UART_AT_CMD_CHAR_DET_INT_RAW UART_RS485_CLASH_INT_RAW UART_RS485_FRM_ERR_INT_RAW UART_TX_DONE_INT_RAW UART_TX_BRK_IDLE_DONE_INT_RAW UART_GLITCH_DET_INT_RAW UART_SW_XOFF_INT_RAW UART_RXFIFO_TOUT_INT_RAW UART_CTS_CHG_INT_RAW UART_DSR_CHG_INT_RAW UART_FRM_ERR_INT_RAW UART_PARITY_ERR_INT_RAW UART_TXFIFO_EMPTY_INT_RAW UART_RXFIFO_FULL_INT_RAW																		
31													20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							

**UART\_RXFIFO\_FULL\_INT\_RAW** This interrupt raw bit turns to high level when receiver receives more data than what UART\_RXFIFO\_FULL\_THRHD specifies. (RO)

**UART\_TXFIFO\_EMPTY\_INT\_RAW** This interrupt raw bit turns to high level when the amount of data in TX FIFO is less than what UART\_TXFIFO\_EMPTY\_THRHD specifies. (RO)

**UART\_PARITY\_ERR\_INT\_RAW** This interrupt raw bit turns to high level when receiver detects a parity error in the data. (RO)

**UART\_FRM\_ERR\_INT\_RAW** This interrupt raw bit turns to high level when receiver detects a data frame error. (RO)

**UART\_RXFIFO\_OVF\_INT\_RAW** This interrupt raw bit turns to high level when receiver receives more data than the FIFO can store. (RO)

**UART\_DSR\_CHG\_INT\_RAW** This interrupt raw bit turns to high level when receiver detects the edge change of DSRn signal. (RO)

**UART\_CTS\_CHG\_INT\_RAW** This interrupt raw bit turns to high level when receiver detects the edge change of CTSn signal. (RO)

**UART\_BRK\_DET\_INT\_RAW** This interrupt raw bit turns to high level when receiver detects a 0 after the stop bit. (RO)

**UART\_RXFIFO\_TOUT\_INT\_RAW** This interrupt raw bit turns to high level when receiver takes more time than UART\_RX\_TOUT\_THRHD to receive a byte. (RO)

**UART\_SW\_XON\_INT\_RAW** This interrupt raw bit turns to high level when receiver receives XON character when UART\_SW\_FLOW\_CON\_EN is set to 1. (RO)

**UART\_SW\_XOFF\_INT\_RAW** This interrupt raw bit turns to high level when receiver receives XOFF character when UART\_SW\_FLOW\_CON\_EN is set to 1. (RO)

**UART\_GLITCH\_DET\_INT\_RAW** This interrupt raw bit turns to high level when receiver detects a glitch in the middle of a start bit. (RO)

**UART\_TX\_BRK\_DONE\_INT\_RAW** This interrupt raw bit turns to high level when transmitter completes sending NULL characters, after all data in TX FIFO are sent. (RO)

**UART\_TX\_BRK\_IDLE\_DONE\_INT\_RAW** This interrupt raw bit turns to high level when transmitter has kept the shortest duration after sending the last data. (RO)

Continued on the next page...

**Register 8.3: UART\_INT\_RAW\_REG (0x0004)**

Continued from the previous page...

**UART\_TX\_DONE\_INT\_RAW** This interrupt raw bit turns to high level when transmitter has sent out all data in FIFO. (RO)

**UART\_RS485\_PARITY\_ERR\_INT\_RAW** This interrupt raw bit turns to high level when receiver detects a parity error from the echo of transmitter in RS485 mode. (RO)

**UART\_RS485\_FRM\_ERR\_INT\_RAW** This interrupt raw bit turns to high level when receiver detects a data frame error from the echo of transmitter in RS485 mode. (RO)

**UART\_RS485\_CLASH\_INT\_RAW** This interrupt raw bit turns to high level when detects a clash between transmitter and receiver in RS485 mode. (RO)

**UART\_AT\_CMD\_CHAR\_DET\_INT\_RAW** This interrupt raw bit turns to high level when receiver detects the configured UART\_AT\_CMD CHAR. (RO)

**UART\_WAKEUP\_INT\_RAW** This interrupt raw bit turns to high level when input rxd edge changes more times than what UART\_ACTIVE\_THRESHOLD specifies in Light-sleep mode. (RO)

**Register 8.4: UART\_INT\_ST\_REG (0x0008)**

(reserved)																UART_WAKEUP_INT_ST UART_AT_CMD_CHAR_DET_INT_ST UART_RS485_CLASH_INT_ST UART_RS485_FRM_ERR_INT_ST UART_RS485_PARITY_ERR_INT_ST UART_TX_DONE_INT_ST UART_TX_BRK_IDLE_DONE_INT_ST UART_GLITCH_DET_INT_ST UART_SW_XON_INT_ST UART_SW_XOFF_INT_ST UART_RXFIFO_TOUT_INT_ST UART_CTS_CHG_INT_ST UART_DSR_CHG_INT_ST UART_RXFIFO_OVF_INT_ST UART_FRM_ERR_INT_ST UART_PARITY_ERR_INT_ST UART_TXFIFO_EMPTY_INT_ST UART_RXFIFO_FULL_INT_ST															
31											20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset	

**UART\_RXFIFO\_FULL\_INT\_ST** This is the status bit for UART\_RXFIFO\_FULL\_INT\_RAW when UART\_RXFIFO\_FULL\_INT\_ENA is set to 1. (RO)

**UART\_TXFIFO\_EMPTY\_INT\_ST** This is the status bit for UART\_TXFIFO\_EMPTY\_INT\_RAW when UART\_TXFIFO\_EMPTY\_INT\_ENA is set to 1. (RO)

**UART\_PARITY\_ERR\_INT\_ST** This is the status bit for UART\_PARITY\_ERR\_INT\_RAW when UART\_PARITY\_ERR\_INT\_ENA is set to 1. (RO)

**UART\_FRM\_ERR\_INT\_ST** This is the status bit for UART\_FRM\_ERR\_INT\_RAW when UART\_FRM\_ERR\_INT\_ENA is set to 1. (RO)

**UART\_RXFIFO\_OVF\_INT\_ST** This is the status bit for UART\_RXFIFO\_OVF\_INT\_RAW when UART\_RXFIFO\_OVF\_INT\_ENA is set to 1. (RO)

**UART\_DSR\_CHG\_INT\_ST** This is the status bit for UART\_DSR\_CHG\_INT\_RAW when UART\_DSR\_CHG\_INT\_ENA is set to 1. (RO)

**UART\_CTS\_CHG\_INT\_ST** This is the status bit for UART\_CTS\_CHG\_INT\_RAW when UART\_CTS\_CHG\_INT\_ENA is set to 1. (RO)

**UART\_BRK\_DET\_INT\_ST** This is the status bit for UART\_BRK\_DET\_INT\_RAW when UART\_BRK\_DET\_INT\_ENA is set to 1. (RO)

**UART\_RXFIFO\_TOUT\_INT\_ST** This is the status bit for UART\_RXFIFO\_TOUT\_INT\_RAW when UART\_RXFIFO\_TOUT\_INT\_ENA is set to 1. (RO)

**UART\_SW\_XON\_INT\_ST** This is the status bit for UART\_SW\_XON\_INT\_RAW when UART\_SW\_XON\_INT\_ENA is set to 1. (RO)

**UART\_SW\_XOFF\_INT\_ST** This is the status bit for UART\_SW\_XOFF\_INT\_RAW when UART\_SW\_XOFF\_INT\_ENA is set to 1. (RO)

**UART\_GLITCH\_DET\_INT\_ST** This is the status bit for UART\_GLITCH\_DET\_INT\_RAW when UART\_GLITCH\_DET\_INT\_ENA is set to 1. (RO)

**UART\_TX\_BRK\_DONE\_INT\_ST** This is the status bit for UART\_TX\_BRK\_DONE\_INT\_RAW when UART\_TX\_BRK\_DONE\_INT\_ENA is set to 1. (RO)

**UART\_TX\_BRK\_IDLE\_DONE\_INT\_ST** This is the status bit for UART\_TX\_BRK\_IDLE\_DONE\_INT\_RAW when UART\_TX\_BRK\_IDLE\_DONE\_INT\_ENA is set to 1. (RO)

**Continued on the next page...**



**Register 8.4: UART\_INT\_ST\_REG (0x0008)**

Continued from the previous page...

**UART\_TX\_DONE\_INT\_ST** This is the status bit for UART\_TX\_DONE\_INT\_RAW when UART\_TX\_DONE\_INT\_ENA is set to 1. (RO)

**UART\_RS485\_PARITY\_ERR\_INT\_ST** This is the status bit for UART\_RS485\_PARITY\_ERR\_INT\_RAW when UART\_RS485\_PARITY\_INT\_ENA is set to 1. (RO)

**UART\_RS485\_FRM\_ERR\_INT\_ST** This is the status bit for UART\_RS485\_FRM\_ERR\_INT\_RAW when UART\_RS485\_FM\_ERR\_INT\_ENA is set to 1. (RO)

**UART\_RS485\_CLASH\_INT\_ST** This is the status bit for UART\_RS485\_CLASH\_INT\_RAW when UART\_RS485\_CLASH\_INT\_ENA is set to 1. (RO)

**UART\_AT\_CMD\_CHAR\_DET\_INT\_ST** This is the status bit for UART\_AT\_CMD\_DET\_INT\_RAW when UART\_AT\_CMD\_CHAR\_DET\_INT\_ENA is set to 1. (RO)

**UART\_WAKEUP\_INT\_ST** This is the status bit for UART\_WAKEUP\_INT\_RAW when UART\_WAKEUP\_INT\_ENA is set to 1. (RO)

Register 8.5: UART\_INT\_ENA\_REG (0x000C)

(reserved)																															UART_WAKEUP_INT_ENA UART_AT_CMD_CHAR_DET_INT_ENA UART_RS485_CLASH_INT_ENA UART_RS485_FRM_ERR_INT_ENA UART_RS485_PARITY_ERR_INT_ENA UART_TX_DONE_INT_ENA UART_TX_BRK_IDLE_DONE_INT_ENA UART_GLITCH_DET_INT_ENA UART_SW_XOFF_INT_ENA UART_SW_XON_INT_ENA UART_RXFIFO_TOUT_INT_ENA UART_CTS_CHG_INT_ENA UART_DSR_CHG_INT_ENA UART_RXFIFO_OVF_INT_ENA UART_FRM_ERR_INT_ENA UART_PARITY_ERR_INT_ENA UART_TXFIFO_EMPTY_INT_ENA UART_RXFIFO_FULL_INT_ENA																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																								20												19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
0												0												0												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Reset

**UART\_RXFIFO\_FULL\_INT\_ENA** This is the enable bit for UART\_RXFIFO\_FULL\_INT\_ST register.

(R/W)

**UART\_TXFIFO\_EMPTY\_INT\_ENA** This is the enable bit for UART\_TXFIFO\_EMPTY\_INT\_ST register.

(R/W)

**UART\_PARITY\_ERR\_INT\_ENA** This is the enable bit for UART\_PARITY\_ERR\_INT\_ST register. (R/W)

**UART\_FRM\_ERR\_INT\_ENA** This is the enable bit for UART\_FRM\_ERR\_INT\_ST register. (R/W)

**UART\_RXFIFO\_OVF\_INT\_ENA** This is the enable bit for UART\_RXFIFO\_OVF\_INT\_ST register. (R/W)

**UART\_DSR\_CHG\_INT\_ENA** This is the enable bit for UART\_DSR\_CHG\_INT\_ST register. (R/W)

**UART\_CTS\_CHG\_INT\_ENA** This is the enable bit for UART\_CTS\_CHG\_INT\_ST register. (R/W)

**UART\_BRK\_DET\_INT\_ENA** This is the enable bit for UART\_BRK\_DET\_INT\_ST register. (R/W)

**UART\_RXFIFO\_TOUT\_INT\_ENA** This is the enable bit for UART\_RXFIFO\_TOUT\_INT\_ST register.

(R/W)

**UART\_SW\_XON\_INT\_ENA** This is the enable bit for UART\_SW\_XON\_INT\_ST register. (R/W)

**UART\_SW\_XOFF\_INT\_ENA** This is the enable bit for UART\_SW\_XOFF\_INT\_ST register. (R/W)

**UART\_GLITCH\_DET\_INT\_ENA** This is the enable bit for UART\_GLITCH\_DET\_INT\_ST register. (R/W)

**UART\_TX\_BRK\_DONE\_INT\_ENA** This is the enable bit for UART\_TX\_BRK\_DONE\_INT\_ST register.

(R/W)

**UART\_TX\_BRK\_IDLE\_DONE\_INT\_ENA** This is the enable bit for UART\_TX\_BRK\_IDLE\_DONE\_INT\_ST register. (R/W)

**UART\_TX\_DONE\_INT\_ENA** This is the enable bit for UART\_TX\_DONE\_INT\_ST register. (R/W)

**UART\_RS485\_PARITY\_ERR\_INT\_ENA** This is the enable bit for UART\_RS485\_PARITY\_ERR\_INT\_ST register. (R/W)

**UART\_RS485\_FRM\_ERR\_INT\_ENA** This is the enable bit for UART\_RS485\_FRM\_ERR\_INT\_ST register. (R/W)

Continued on the next page...

**Register 8.5: UART\_INT\_ENA\_REG (0x000C)**

Continued from the previous page...

**UART\_RS485\_CLASH\_INT\_ENA** This is the enable bit for UART\_RS485\_CLASH\_INT\_ST register.  
(R/W)

**UART\_AT\_CMD\_CHAR\_DET\_INT\_ENA** This is the enable bit for  
UART\_AT\_CMD\_CHAR\_DET\_INT\_ST register. (R/W)

**UART\_WAKEUP\_INT\_ENA** This is the enable bit for UART\_WAKEUP\_INT\_ST register. (R/W)

Register 8.6: UART\_INT\_CLR\_REG (0x0010)

(reserved)																				UART_WAKEUP_INT_CLR UART_AT_CMD_CHAR_DET_INT_CLR UART_RS485_CLASH_INT_CLR UART_RS485_FRM_ERR_INT_CLR UART_TX_DONE_INT_CLR UART_TX_BRK_IDLE_DONE_INT_CLR UART_GLITCH_DET_INT_CLR UART_SW_XOFF_INT_CLR UART_SW_XON_INT_CLR UART_BRK_DET_INT_CLR UART_CTS_CHG_INT_CLR UART_DSR_CHG_INT_CLR UART_RXFIFO_OVF_INT_CLR UART_FRM_ERR_INT_CLR UART_PARITY_ERR_INT_CLR UART_TXFIFO_EMPTY_INT_CLR UART_RXFIFO_FULL_INT_CLR																					
31																				20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													

**UART\_RXFIFO\_FULL\_INT\_CLR** Set this bit to clear UART\_THE\_RXFIFO\_FULL\_INT\_RAW interrupt. (WO)

**UART\_TXFIFO\_EMPTY\_INT\_CLR** Set this bit to clear UART\_TXFIFO\_EMPTY\_INT\_RAW interrupt. (WO)

**UART\_PARITY\_ERR\_INT\_CLR** Set this bit to clear UART\_PARITY\_ERR\_INT\_RAW interrupt. (WO)

**UART\_FRM\_ERR\_INT\_CLR** Set this bit to clear UART\_FRM\_ERR\_INT\_RAW interrupt. (WO)

**UART\_RXFIFO\_OVF\_INT\_CLR** Set this bit to clear UART\_UART\_RXFIFO\_OVF\_INT\_RAW interrupt. (WO)

**UART\_DSR\_CHG\_INT\_CLR** Set this bit to clear UART\_DSR\_CHG\_INT\_RAW interrupt. (WO)

**UART\_CTS\_CHG\_INT\_CLR** Set this bit to clear UART\_CTS\_CHG\_INT\_RAW interrupt. (WO)

**UART\_BRK\_DET\_INT\_CLR** Set this bit to clear UART\_BRK\_DET\_INT\_RAW interrupt. (WO)

**UART\_RXFIFO\_TOUT\_INT\_CLR** Set this bit to clear UART\_RXFIFO\_TOUT\_INT\_RAW interrupt. (WO)

**UART\_SW\_XON\_INT\_CLR** Set this bit to clear UART\_SW\_XON\_INT\_RAW interrupt. (WO)

**UART\_SW\_XOFF\_INT\_CLR** Set this bit to clear UART\_SW\_XOFF\_INT\_RAW interrupt. (WO)

**UART\_GLITCH\_DET\_INT\_CLR** Set this bit to clear UART\_GLITCH\_DET\_INT\_RAW interrupt. (WO)

**UART\_TX\_BRK\_DONE\_INT\_CLR** Set this bit to clear UART\_TX\_BRK\_DONE\_INT\_RAW interrupt. (WO)

**UART\_TX\_BRK\_IDLE\_DONE\_INT\_CLR** Set this bit to clear UART\_TX\_BRK\_IDLE\_DONE\_INT\_RAW interrupt. (WO)

**UART\_TX\_DONE\_INT\_CLR** Set this bit to clear UART\_TX\_DONE\_INT\_RAW interrupt. (WO)

**UART\_RS485\_PARITY\_ERR\_INT\_CLR** Set this bit to clear UART\_RS485\_PARITY\_ERR\_INT\_RAW interrupt. (WO)

**UART\_RS485\_FRM\_ERR\_INT\_CLR** Set this bit to clear UART\_RS485\_FRM\_ERR\_INT\_RAW interrupt. (WO)

Continued on the next page...

Register 8.6: UART\_INT\_CLR\_REG (0x0010)

Continued from the previous page...

**UART\_RS485\_CLASH\_INT\_CLR** Set this bit to clear UART\_RS485\_CLASH\_INT\_RAW interrupt. (WO)

**UART\_AT\_CMD\_CHAR\_DET\_INT\_CLR** Set this bit to clear UART\_AT\_CMD\_CHAR\_DET\_INT\_RAW interrupt. (WO)

**UART\_WAKEUP\_INT\_CLR** Set this bit to clear UART\_WAKEUP\_INT\_RAW interrupt. (WO)

Register 8.7: UART\_CLKDIV\_REG (0x0014)

(reserved)								UART_CLKDIV_FRAG				UART_CLKDIV															
31								24	23	20	19													0			
0	0	0	0	0	0	0	0	0	0x0		0x2b6														Reset		

**UART\_CLKDIV** The integral part of the frequency divisor. (R/W)

**UART\_CLKDIV\_FRAG** The decimal part of the frequency divisor. (R/W)

**Register 8.8: UART\_CONF0\_REG (0x0020)**

<div>(reserved)</div>																															UART_MEM_CLK_EN		UART_TICK_REF_ALWAYS_ON		<div>(reserved)</div>			UART_DTR_INV		UART_RTS_INV		UART_TXD_INV		UART_DSR_INV		UART_CTS_INV		UART_RXD_INV		UART_TXFIFO_RST		UART_RXFIFO_RST		UART_IRDA_EN		UART_TX_FLOW_EN		UART_LOOPBACK		UART_IRDA_RX_INV		UART_IRDA_TX_INV		UART_IRDA_WCTL		UART_IRDA_TX_EN		UART_TXD_DPLX		UART_SW_BRK		UART_SW_DTR		UART_STOP_BIT_NUM		UART_BIT_NUM		UART_PARITY_EN		UART_PARITY	
31	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	Reset																																																		
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0																																																				

**UART\_PARITY** This register is used to configure the parity check mode. 1'h0: even. 1'h1: odd.  
(R/W)

**UART\_PARITY\_EN** Set this bit to enable UART parity check. (R/W)

**UART\_BIT\_NUM** This register is used to set the length of data. 0: 5 bits 1: 6 bits 2: 7 bits 3: 8 bits.  
(R/W)

**UART\_STOP\_BIT\_NUM** This register is used to set the length of stop bit. 1: 1 bit 2: 1.5 bits 3: 2 bits. (R/W)

**UART\_SW\_RTS** This register is used to configure the software RTS signal which is used in software flow control. (R/W)

**UART\_SW\_DTR** This register is used to configure the software DTR signal which is used in software flow control. (R/W)

**UART\_TXD\_BRK** Set this bit to enable transmitter to send NULL when the process of sending data is done. (R/W)

**UART\_IRDA\_DPLX** Set this bit to enable IrDA loopback mode. (R/W)

**UART\_IRDA\_TX\_EN** This is the start enable bit for IrDA transmitter. (R/W)

**UART\_IRDA\_WCTL** 1'h1: The IrDA transmitter's 11th bit is the same as 10th bit. 1'h0: Set IrDA transmitter's 11th bit to 0. (R/W)

**UART\_IRDA\_TX\_INV** Set this bit to invert the level of IrDA transmitter. (R/W)

**UART\_IRDA\_RX\_INV** Set this bit to invert the level of IrDA receiver. (R/W)

**UART\_LOOPBACK** Set this bit to enable UART loopback test mode. (R/W)

**UART\_TX\_FLOW\_EN** Set this bit to enable flow control function for transmitter. (R/W)

**UART\_IRDA\_EN** Set this bit to enable IrDA protocol. (R/W)

**UART\_RXFIFO\_RST** Set this bit to reset the UART RX FIFO. (R/W)

**UART\_TXFIFO\_RST** Set this bit to reset the UART TX FIFO. (R/W)

**UART\_RXD\_INV** Set this bit to inverse the level value of UART RXD signal. (R/W)

**UART\_CTS\_INV** Set this bit to inverse the level value of UART CTS signal. (R/W)

Continued on the next page...

**Register 8.8: UART\_CONF0\_REG (0x0020)**

Continued from the previous page...

**UART\_DSR\_INV** Set this bit to inverse the level value of UART DSR signal. (R/W)**UART\_TXD\_INV** Set this bit to inverse the level value of UART TXD signal. (R/W)**UART\_RTS\_INV** Set this bit to inverse the level value of UART RTS signal. (R/W)**UART\_DTR\_INV** Set this bit to inverse the level value of UART DTR signal. (R/W)**UART\_TICK\_REF\_ALWAYS\_ON** This register is used to select the clock. 1'h1: APB\_CLK. 1'h0: REF\_TICK. (R/W)**UART\_MEM\_CLK\_EN** UART memory clock gate enable signal. 1: UART memory powers on, the data of which can be read and written. 0: UART memory powers down. (R/W)**Register 8.9: UART\_CONF1\_REG (0x0024)**

UART_RX_TOUT_EN																(reserved)																UART_TXFIFO_EMPTY_THRHD																UART_RXFIFO_FULL_THRHD															
UART_RX_FLOW_EN																																Reset																															
UART_RX_TOUT_FLOW_DIS																																																															
31	30	29	28													18	17													9	8													0																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x60												0x60												Reset																							

Reset

**UART\_RXFIFO\_FULL\_THRHD** It will produce UART\_RXFIFO\_FULL\_INT interrupt when receiver receives more data than this register value. (R/W)**UART\_TXFIFO\_EMPTY\_THRHD** It will produce UART\_TXFIFO\_EMPTY\_INT interrupt when the data amount in TX FIFO is less than this register value. (R/W)**UART\_RX\_TOUT\_FLOW\_DIS** Set this bit to stop accumulating idle\_cnt when hardware flow control works. (R/W)**UART\_RX\_FLOW\_EN** This is the flow enable bit for UART receiver. 1'h1: Choose software flow control with configuring sw\_rts signal. 1'h0: Disable software flow control. (R/W)**UART\_RX\_TOUT\_EN** This is the enable bit for UART receiver's timeout function. (R/W)

Register 8.10: UART\_FLOW\_CONF\_REG (0x0034)

(reserved)																								UART_SEND_XOFF			
																								UART_SEND_XON			
																								UART_FORCE_XOFF			
																								UART_FORCE_XON			
																								UART_XONOFF_DEL			
																								UART_SW_FLOW_CON_EN			
31																			6	5	4	3	2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

- UART\_SW\_FLOW\_CON\_EN** Set this bit to enable software flow control. It is used with register SW\_XON or SW\_XOFF. (R/W)
- UART\_XONOFF\_DEL** Set this bit to remove flow control character from the received data. (R/W)
- UART\_FORCE\_XON** Set this bit to enable the transmitter to go on sending data. (R/W)
- UART\_FORCE\_XOFF** Set this bit to stop the transmitter from sending data. (R/W)
- UART\_SEND\_XON** Set this bit to send XON character. It is cleared by hardware automatically. (R/W)
- UART\_SEND\_XOFF** Set this bit to send XOFF character. It is cleared by hardware automatically. (R/W)

Register 8.11: UART\_SLEEP\_CONF\_REG (0x0038)

										(reserved)																				UART_ACTIVE_THRESHOLD									
31										10										9										0									
0 0										0xf0										Reset																			

- UART\_ACTIVE\_THRESHOLD** The UART is activated from light sleeping mode when the input rxd edge changes more times than this register value. (R/W)



**Register 8.12: UART\_SWFC\_CONF0\_REG (0x003C)**

(reserved)																UART_XOFF_CHAR								UART_XOFF_THRESHOLD																	
31																17								16								9	8								0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x13							0xe0							Reset											

**UART\_XOFF\_THRESHOLD** When the data amount in RX FIFO is more than this register value with UART\_SW\_FLOW\_CON\_EN set to 1, it will send a XOFF character. (R/W)

**UART\_XOFF\_CHAR** This register stores the XOFF flow control character. (R/W)

**Register 8.13: UART\_SWFC\_CONF1\_REG (0x0040)**

(reserved)																UART_XON_CHAR								UART_XON_THRESHOLD										
31																17	16								9	8								0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x11							0x0							Reset				

**UART\_XON\_THRESHOLD** When the data amount in RX FIFO is less than this register value with UART\_SW\_FLOW\_CON\_EN set to 1, it will send a XON character. (R/W)

**UART\_XON\_CHAR** This register stores the XON flow control character. (R/W)

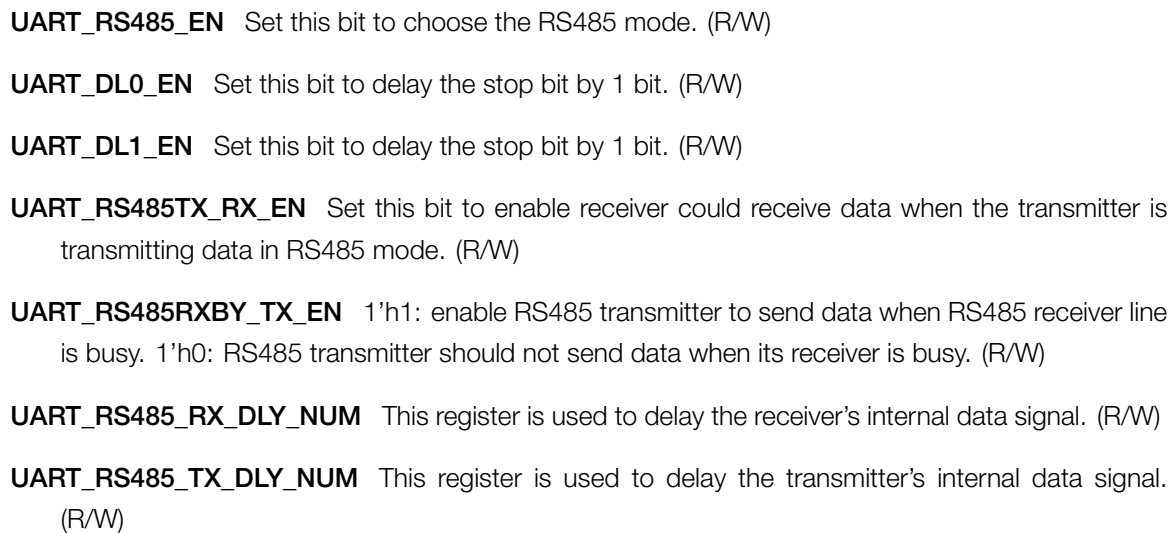
**Register 8.14: UART\_IDLE\_CONF\_REG (0x0044)**

(reserved)				UART_TX_BRK_NUM																UART_TX_IDLE_NUM																UART_RX_IDLE_THRHD															
31				28				27				20				19				10				9				0																							
0				0				0				0				0xa				0x100				0x100				0x100				Reset																			

**UART\_RX\_IDLE\_THRHD** It will produce frame end signal when receiver takes more time to receive one byte data than this register value. (R/W)

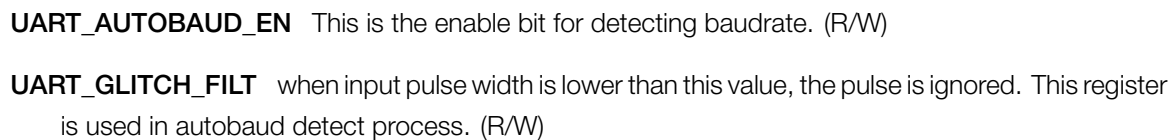
**UART\_TX\_IDLE\_NUM** This register is used to configure the duration time between transfers. (R/W)

**UART\_TX\_BRK\_NUM** This register is used to configure the number of 0 to be sent after the process of sending data is done. It is active when UART\_TXD\_BRK is set to 1. (R/W)

[Submit Documentation Feedback](#)

**UART\_AUTOBAUD\_EN** This is the enable bit for detecting baudrate. (R/W)

**UART\_GLITCH\_FILTER** when input pulse width is lower than this value, the pulse is ignored. This register is used in autobaud detect process. (R/W)

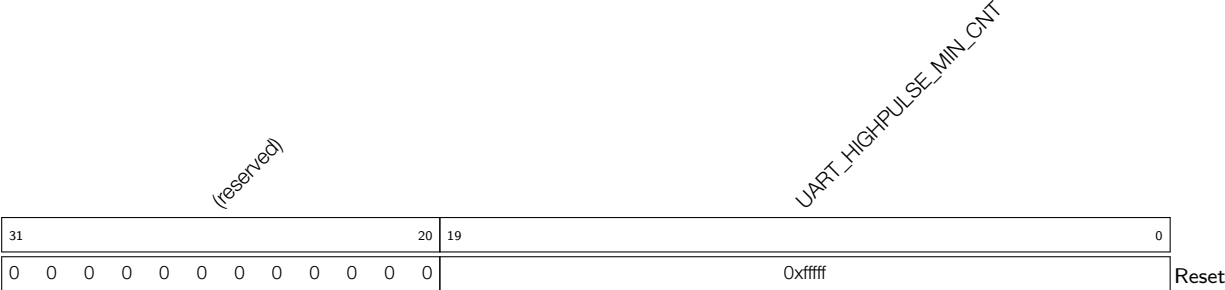


Register 8.17: UART\_LOWPULSE\_REG (0x0028)



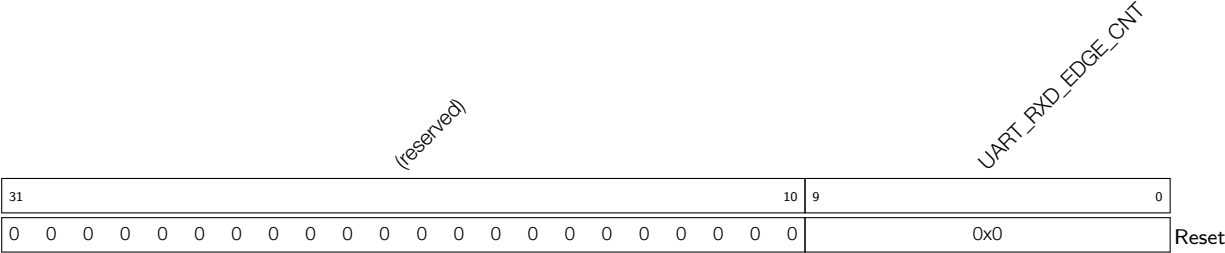
**UART\_LOWPULSE\_MIN\_CNT** This register stores the value of the minimum duration time of the low level pulse. It is used in baud rate detection. (RO)

Register 8.18: UART\_HIGHPULSE\_REG (0x002C)



**UART\_HIGHPULSE\_MIN\_CNT** This register stores the value of the maximum duration time for the high level pulse. It is used in baud rate detection. (RO)

Register 8.19: UART\_RXD\_CNT\_REG (0x0030)



**UART\_RXD\_EDGE\_CNT** This register stores the count of rxd edge change. It is used in baud rate detection. (RO)

### Register 8.20: UART\_POSPULSE\_REG (0x006C)

Diagram illustrating the structure of the UART\_PCR register. The register is 32 bits wide, divided into two 16-bit halves. The top half (bits 31-16) is labeled "(reserved)". The bottom half (bits 15-0) is labeled "UART\_POSEDGE\_MIN\_CNT". The bottom half is further divided into a 16-bit field labeled "0xffff" and a 1-bit field labeled "Reset".

**UART\_POSEDGE\_MIN\_CNT** This register stores the minimal input clock count between two positive edges. It is used in baud rate detection. (RO)

### Register 8.21: UART NEGPULSE REG (0x0070)

Diagram illustrating the structure of the `UART_NEDGE_MIN_CNT` register. The register is 32 bits wide, divided into two 16-bit halves. The top half (bits 31-16) is labeled `(reserved)`. The bottom half (bits 15-0) is labeled `UART_NEDGE_MIN_CNT`. The bottom half is further divided into a 16-bit field labeled `0xf` and a 16-bit field labeled `Reset`.

**UART\_NEGEDGE\_MIN\_CNT** This register stores the minimal input clock count between two negative edges. It is used in baud rate detection. (RO)

**Register 8.22: UART\_STATUS\_REG (0x001C)**

UART_TXD UART_RTSN UART_DTRN (reserved)						UART_TXFIFO_CNT						UART_RXD UART_CTSN UART_DSRN (reserved)						UART_RXFIFO_CNT						
31	30	29	28	26	25	16	15	14	13	12	10	9	0											
0x0	0	0	0	0	0	0x0						0	0	0	0	0	0	0x0						Reset

Reset

**UART\_RXFIFO\_CNT** Stores the byte number of valid data in RX FIFO. (RO)**UART\_DSRN** The register represent the level value of the internal UART DSR signal. (RO)**UART\_CTSN** This register represent the level value of the internal UART CTS signal. (RO)**UART\_RXD** This register represent the level value of the internal UART RXD signal. (RO)**UART\_TXFIFO\_CNT** Stores the byte number of data in TX FIFO. (RO)**UART\_DTRN** This bit represents the level of the internal UART DTR signal. (RO)**UART\_RTSN** This bit represents the level of the internal UART RTS signal. (RO)**UART\_TXD** This bit represents the level of the internal UART TXD signal. (RO)**Register 8.23: UART\_MEM\_TX\_STATUS\_REG (0x0060)**

(reserved)												UART_TX_RADDR												(reserved)												UART_APB_TX_WADDR																																			
31												21												11												10												9												0											
0 0 0 0 0 0 0 0 0 0 0 0												0x0												0												0x0												Reset																							

Reset

**UART\_APB\_TX\_WADDR** This register stores the offset address in TX FIFO when software writes TX FIFO via APB. (RO)**UART\_TX\_RADDR** This register stores the offset address in TX FIFO when TX FSM reads data via Tx\_FIFO\_Ctrl. (RO)

**Register 8.24: UART\_MEM\_RX\_STATUS\_REG (0x0064)**

(reserved)												UART_RX_WADDR												(reserved)												UART_APB_RX_RADDR																																																																																															
31												21												20												11												10												9												0																																																											
0												0												0												0												0												0												0												0x0												0												0x0												Reset											

**UART\_APB\_RX\_RADDR** This register stores the offset address in RX\_FIFO when software reads data from RX FIFO via APB. (RO)

**UART\_RX\_WADDR** This register stores the offset address in RX FIFO when Rx\_FIFO\_Ctrl writes RX FIFO. (RO)

**Register 8.25: UART\_FSM\_STATUS\_REG (0x0068)**

(reserved)																								UART_ST_UTX_OUT								UART_ST_URX_OUT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
31																								8	7	4				3	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**UART\_ST\_URX\_OUT** This is the status register of receiver. (RO)

**UART\_ST\_UTX\_OUT** This is the status register of transmitter. (RO)

**Register 8.26: UART\_AT\_CMD\_PRECNT\_REG (0x004C)**

(reserved)																UART_PRE_IDLE_NUM																															
31																15																0															
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x901																Reset															

**UART\_PRE\_IDLE\_NUM** This register is used to configure the idle duration time before the first AT\_CMD is received by receiver. It will not take the next data received as AT\_CMD character when the duration is less than this register value. (R/W)

## 239

[Submit Documentation Feedback](#)

## 239

[Submit Documentation Feedback](#)

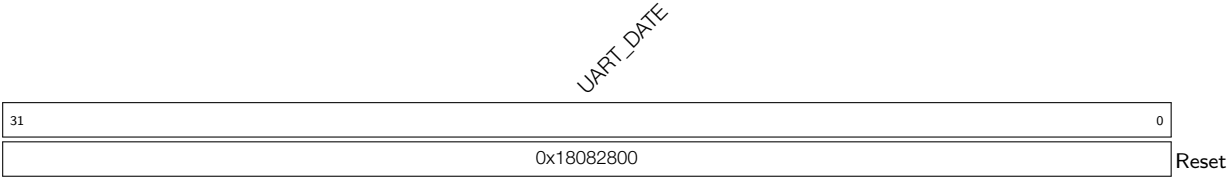
## 239



UART\_AT\_CMD\_CHAR

ESP32-S2 TRM (Preliminary V0.5)

Register 8.30: UART\_DATE\_REG (0x0074)



**UART\_DATE** This is the version control register. (R/W)



### Register 8.31: UHCI CONF0 REG (0x0000)

[illegible]

**UHCI\_IN\_RST** Set this bit to reset in DMA FSM. (R/W)

**UHCI\_OUT\_RST** Set this bit to reset out DMA FSM. (R/W)

**UHCI\_AHBM\_FIFO\_RST** Set this bit to reset AHB interface cmdFIFO of DMA. (R/W)

**UHCI\_AHBM\_RST** Set this bit to reset AHB interface of DMA. (R/W)

**UHCI\_IN\_LOOP\_TEST** Reserved. (R/W)

**UHCI\_OUT\_LOOP\_TEST** Reserved. (R/W)

**UHCI\_OUT\_AUTO\_WBACK** Set this bit to enable automatic outlink-writeback when all the data in TX Buffer has been transmitted. (R/W)

**UHCI\_OUT\_NO\_RESTART\_CLR** Reserved. (R/W)

**UHCI\_OUT\_EOF\_MODE** This register is used to specify the generation mode of UHCI\_OUT\_EOF\_INT interrupt. 1: When DMA has popped all data from FIFO. 0: When AHB has pushed all data to FIFO. (R/W)

**UHCI\_UART0\_CE** Set this bit to link up HCI and UART0. (R/W)

**UHCI\_UART1\_CE** Set this bit to link up HCI and UART1. (R/W)

**UHCI\_OUTDSR\_BURST\_EN** This register is used to specify DMA transmit descriptor transfer mode. 1: burst mode. 0: byte mode. (R/W)

**UHCI\_INDSCR\_BURST\_EN** This register is used to specify DMA receive descriptor transfer mode.  
1: burst mode. 0: byte mode. (R/W)

**UHCI\_MEM\_TRANS\_EN** 1: UHCI transmitted data would be write back into DMA INFIFO. (R/W)

**UHCI\_SEPER\_EN** Set this bit to separate the data frame using a special char. (R/W)

**UHCI\_HEAD\_EN** Set this bit to encode the data packet with a formatting header. (R/W)

**UHCI\_CRC\_REC\_EN** Set this bit to enable UHCI to receive the 16 bit CRC. (R/W)

**UHCI\_UART\_IDLE\_EOF\_EN** If this bit is set to 1, UHCI will end the payload receiving process when UART has been in idle state. (R/W)

Continued on the next page...

**Register 8.31: UHCI\_CONF0\_REG (0x0000)**

**Continued from the previous page...**

**UHCI\_LEN\_EOF\_EN** If this bit is set to 1, UHCI decoder receiving payload data is end when the receiving byte count has reached the specified value. The value is payload length indicated by UCHI packet header when UHCI\_HEAD\_EN is 1 or the value is a configuration value when UHCI\_HEAD\_EN is 0. If this bit is set to 0, UHCI decoder receiving payload data is end when 0xc0 is received. (R/W)

**UHCI\_ENCODE\_CRC\_EN** Set this bit to enable data integrity checking by appending a 16 bit CCITT-CRC to the end of the payload. (R/W)

**UHCI\_CLK\_EN** 1'b1: Force clock on for register. 1'b0: Support clock only when application writes registers. (R/W)

**UHCI\_UART\_RX\_BRK\_EOF\_EN** if this bit is set to 1, UHCI will end payload\_rec process when NULL frame is received by UART. (R/W)

**Register 8.32: UHCI\_CONF1\_REG (0x002C)**

(reserved)												UHCI_DMA_INFIFO_FULL_THRS												UHCI_SW_START UHCI_WAIT_SW_START UHCI_CHECK_OWNER UHCI_TX_ACK_NUM_RE UHCI_SAVE_HEAD UHCI_CRC_DISABLE UHCI_CHECK_SEQ_EN UHCI_CHECK_SUM_EN																																	
31												21												20												9												8	7	6	5	4	3	2	1	0	
0												0												0												0	0	0	1	1	0	0	1	1	Reset												

**UHCI\_CHECK\_SUM\_EN** This is the enable bit to check header checksum when UHCI receives a data packet. (R/W)

**UHCI\_CHECK\_SEQ\_EN** This is the enable bit to check sequence number when UHCI receives a data packet. (R/W)

**UHCI\_CRC\_DISABLE** Set this bit to support CRC calculation. Data Integrity Check Present bit in UHCI packet frame should be 1. (R/W)

**UHCI\_SAVE\_HEAD** Set this bit to save the packet header when HCI receives a data packet. (R/W)

**UHCI\_TX\_CHECK\_SUM\_RE** Set this bit to encode the data packet with a checksum. (R/W)

**UHCI\_TX\_ACK\_NUM\_RE** Set this bit to encode the data packet with an acknowledgement when a reliable packet is to be transmit. (R/W)

**UHCI\_CHECK\_OWNER** 1: Check the link list descriptor when link list owner is DMA controller; 0: Always check link list descriptor. (R/W)

**UHCI\_WAIT\_SW\_START** The uhci-encoder will jump to ST\_SW\_WAIT status if this register is set to 1. (R/W)

**UHCI\_SW\_START** If current UHCI\_ENCODE\_STATE is ST\_SW\_WAIT, the UHCI will start to send data packet out when this bit is set to 1. (R/W)

**UHCI\_DMA\_INFIFO\_FULL\_THRS** This field is used to generate the UHCI\_DMA\_INFIFO\_FULL\_WM\_INT interrupt when the counter value of DMA RX FIFO exceeds the value of the register. (R/W)

**Register 8.33: UHCI\_AHB\_TEST\_REG (0x0048)**

(reserved)																										UHCI_AHB_TESTADDR (reserved)		UHCI_AHB_TESTMODE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																										6	5	4	3	2	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

**UHCI\_AHB\_TESTMODE** Reserved. (R/W)

**UHCI\_AHB\_TESTADDR** Reserved. (R/W)

**Register 8.34: UHCI\_ESCAPE\_CONF\_REG (0x0064)**

(reserved)																								UHCL_RX_13_ESC_EN								UHCL_RX_11_ESC_EN								UHCL_RX_DB_ESC_EN								UHCL_RX_C0_ESC_EN								UHCL_TX_13_ESC_EN								UHCL_TX_11_ESC_EN								UHCL_TX_DB_ESC_EN								UHCL_TX_C0_ESC_EN							
31																								8								7	6	5	4	3	2	1	0																																																
0																								0								0	0	1	1	0	0	1	1	Reset																																															

**UHCI\_TX\_C0\_ESC\_EN** Set this bit to enable decoding char 0xc0 when DMA receives data. (R/W)

**UHCI\_TX\_DB\_ESC\_EN** Set this bit to enable decoding char 0xdb when DMA receives data. (R/W)

**UHCI\_TX\_11\_ESC\_EN** Set this bit to enable decoding flow control char 0x11 when DMA receives data. (R/W)

**UHCI\_TX\_13\_ESC\_EN** Set this bit to enable decoding flow control char 0x13 when DMA receives data. (R/W)

**UHCI\_RX\_C0\_ESC\_EN** Set this bit to enable replacing 0xc0 by special char when DMA sends data. (R/W)

**UHCI\_RX\_DB\_ESC\_EN** Set this bit to enable replacing 0xdb by special char when DMA sends data. (R/W)

**UHCI\_RX\_11\_ESC\_EN** Set this bit to enable replacing flow control char 0x11 by special char when DMA sends data. (R/W)

**UHCI\_RX\_13\_ESC\_EN** Set this bit to enable replacing flow control char 0x13 by special char when DMA sends data. (R/W)

**Register 8.35: UHCI\_HUNG\_CONF\_REG (0x0068)**

(reserved)								UHCL_RXFIFO_TIMEOUT_ENA				UHCL_RXFIFO_TIMEOUT_SHIFT				UHCL_RXFIFO_TIMEOUT				UHCL_TXFIFO_TIMEOUT_ENA				UHCL_TXFIFO_TIMEOUT_SHIFT				UHCL_TXFIFO_TIMEOUT					
31								24	23	22	20		19								12	11	10	8	7								0
0	0	0	0	0	0	0	0	0	1	0		0x10							1	0	0x10							Reset					

Reset

**UHCI\_TXFIFO\_TIMEOUT** This register stores the timeout value. It will produce the UHCI\_TX\_HUNG\_INT interrupt when DMA takes more time to receive data. (R/W)

**UHCI\_TXFIFO\_TIMEOUT\_SHIFT** This register is used to configure the tick count maximum value. (R/W)

**UHCI\_TXFIFO\_TIMEOUT\_ENA** This is the enable bit for Tx-FIFO receive-data timeout. (R/W)

**UHCI\_RXFIFO\_TIMEOUT** This register stores the timeout value. It will produce the UHCI\_RX\_HUNG\_INT interrupt when DMA takes more time to read data from RAM. (R/W)

**UHCI\_RXFIFO\_TIMEOUT\_SHIFT** This register is used to configure the tick count maximum value. (R/W)

**UHCI\_RXFIFO\_TIMEOUT\_ENA** This is the enable bit for DMA send-data timeout. (R/W)

**Register 8.36: UHCI\_QUICK\_SENT\_REG (0x0074)**

(reserved)																												UHCI_ALWAYS_SEND_EN				UHCI_ALWAYS_SEND_NUM				UHCI_SINGLE_SEND_EN				UHCI_SINGLE_SEND_NUM																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
31																												8	7	6	4				3	2	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

**UHCI\_SINGLE\_SEND\_NUM** This register is used to specify the single\_send register. (R/W)

**UHCI\_SINGLE\_SEND\_EN** Set this bit to enable single\_send mode to send short packet. (R/W)

**UHCI\_ALWAYS\_SEND\_NUM** This register is used to specify the always\_send register. (R/W)

**UHCI\_ALWAYS\_SEND\_EN** Set this bit to enable always\_send mode to send short packet. (R/W)

**Register 8.37: UHCI\_Q0\_WORD0\_REG (0x0078)**

UHCI\_SEND\_Q0\_WORD0

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q0\_WORD0** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

**Register 8.38: UHCI\_Q0\_WORD1\_REG (0x007C)**

UHCI\_SEND\_Q0\_WORD1

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q0\_WORD1** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

**Register 8.39: UHCI\_Q1\_WORD0\_REG (0x0080)**

UHCI\_SEND\_Q1\_WORD0

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q1\_WORD0** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

**Register 8.40: UHCI\_Q1\_WORD1\_REG (0x0084)**

UHCI\_SEND\_Q1\_WORD1

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q1\_WORD1** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

**Register 8.41: UHCI\_Q2\_WORD0\_REG (0x0088)**

UHCI\_SEND\_Q2\_WORD0

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q2\_WORD0** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

**Register 8.42: UHCI\_Q2\_WORD1\_REG (0x008C)**

UHCI\_SEND\_Q2\_WORD1

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q2\_WORD1** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

**Register 8.43: UHCI\_Q3\_WORD0\_REG (0x0090)**

UHCI\_SEND\_Q3\_WORD0

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q3\_WORD0** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

**Register 8.44: UHCI\_Q3\_WORD1\_REG (0x0094)**

UHCI\_SEND\_Q3\_WORD1

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q3\_WORD1** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

**Register 8.45: UHCI\_Q4\_WORD0\_REG (0x0098)**

UHCI\_SEND\_Q4\_WORD0

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q4\_WORD0** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)



**Register 8.46: UHCI\_Q4\_WORD1\_REG (0x009C)**

UHCI\_SEND\_Q4\_WORD1

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q4\_WORD1** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

**Register 8.47: UHCI\_Q5\_WORD0\_REG (0x00A0)**

UHCI\_SEND\_Q5\_WORD0

31	0
0x000000	
Reset	

**UHCI\_SEND\_Q5\_WORD0** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

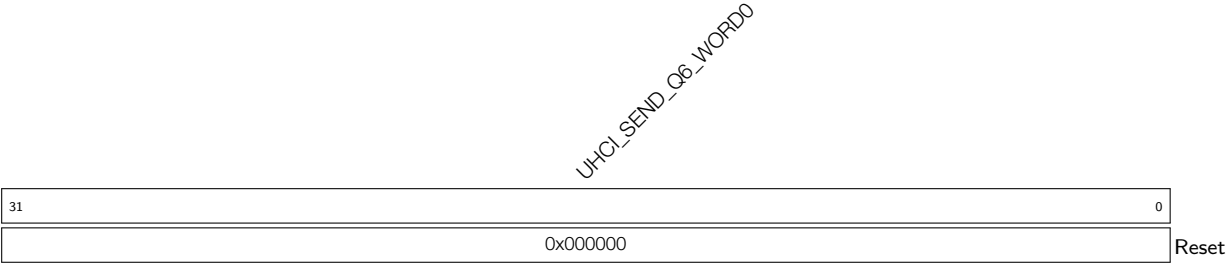
**Register 8.48: UHCI\_Q5\_WORD1\_REG (0x00A4)**

UHCI\_SEND\_Q5\_WORD1

31	0
0x000000	
Reset	

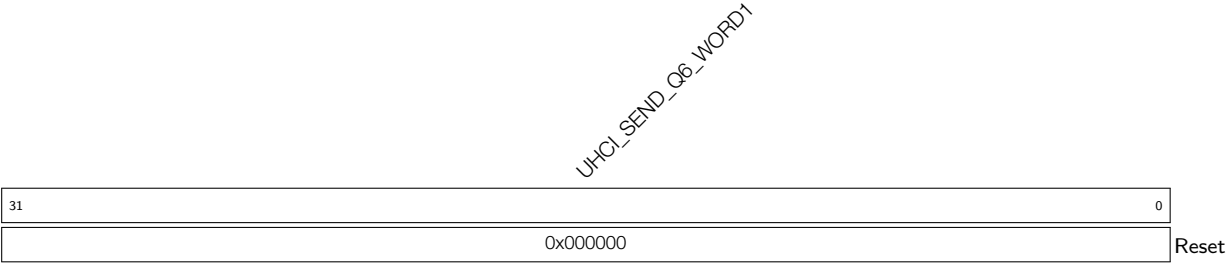
**UHCI\_SEND\_Q5\_WORD1** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

Register 8.49: UHCI\_Q6\_WORD0\_REG (0x00A8)



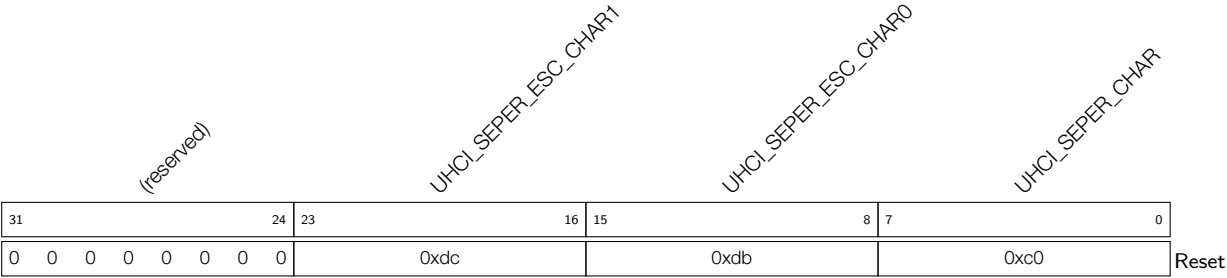
**UHCI\_SEND\_Q6\_WORD0** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

Register 8.50: UHCI\_Q6\_WORD1\_REG (0x00AC)



**UHCI\_SEND\_Q6\_WORD1** This register is used as a quick\_sent register when specified by UHCI\_ALWAYS\_SEND\_NUM or UHCI\_SINGLE\_SEND\_NUM. (R/W)

Register 8.51: UHCI\_ESC\_CONF0\_REG (0x00B0)



**UHCI\_SEPER\_CHAR** This register is used to define the separate char that need to be encoded, default is 0xc0. (R/W)

**UHCI\_SEPER\_ESC\_CHAR0** This register is used to define the first char of slip escape sequence when encoding the separate char, default is 0xdb. (R/W)

**UHCI\_SEPER\_ESC\_CHAR1** This register is used to define the second char of slip escape sequence when encoding the separate char, default is 0xdc. (R/W)

**Register 8.52: UHCI\_ESC\_CONF1\_REG (0x00B4)**

(reserved)								UHCI_ESC_SEQ0_CHAR1								UHCI_ESC_SEQ0_CHAR0								UHCI_ESC_SEQ0							
31	24	23	16	15	8	7	0																								
0	0	0	0	0	0	0	0	0xdd								0xdb								0xdb							

Reset

**UHCI\_ESC\_SEQ0** This register is used to define a char that need to be encoded, default is 0xdb that used as the first char of slip escape sequence. (R/W)

**UHCI\_ESC\_SEQ0\_CHAR0** This register is used to define the first char of slip escape sequence when encoding the UHCI\_ESC\_SEQ0, default is 0xdb. (R/W)

**UHCI\_ESC\_SEQ0\_CHAR1** This register is used to define the second char of slip escape sequence when encoding the UHCI\_ESC\_SEQ0, default is 0xdd. (R/W)

**Register 8.53: UHCI\_ESC\_CONF2\_REG (0x00B8)**

(reserved)								UHCI_ESC_SEQ1_CHAR1								UHCI_ESC_SEQ1_CHAR0								UHCI_ESC_SEQ1							
31	24	23	16	15	8	7	0																								
0	0	0	0	0	0	0	0	0xde								0xdb								0x11							

Reset

**UHCI\_ESC\_SEQ1** This register is used to define a char that need to be encoded, default is 0x11 that used as flow control char. (R/W)

**UHCI\_ESC\_SEQ1\_CHAR0** This register is used to define the first char of slip escape sequence when encoding the UHCI\_ESC\_SEQ1, default is 0xdb. (R/W)

**UHCI\_ESC\_SEQ1\_CHAR1** This register is used to define the second char of slip escape sequence when encoding the UHCI\_ESC\_SEQ1, default is 0xde. (R/W)

**Register 8.54: UHCI\_ESC\_CONF3\_REG (0x00BC)**

(reserved)								UHCI_ESC_SEQ2_CHAR1								UHCI_ESC_SEQ2_CHAR0								UHCI_ESC_SEQ2																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
31								24								23								16								15								8								7								0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0							

**UHCI\_ESC\_SEQ2** This register is used to define a char that need to be decoded, default is 0x13 that used as flow control char. (R/W)

**UHCI\_ESC\_SEQ2\_CHAR0** This register is used to define the first char of slip escape sequence when encoding the UHCI\_ESC\_SEQ2, default is 0xdb. (R/W)

**UHCI\_ESC\_SEQ2\_CHAR1** This register is used to define the second char of slip escape sequence when encoding the UHCI\_ESC\_SEQ2, default is 0xdf. (R/W)

**Register 8.55: UHCI\_PKT\_THRES\_REG (0x00C0)**

(reserved)																UHCL_PKT_THRS																																															
31																13																12																0															
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x80																Reset																															

**UHCI\_PKT\_THRS** This register is used to configure the maximum value of the packet length when UHCI\_HEAD\_EN is 0. (R/W)

**Register 8.56: UHCI\_INT\_RAW\_REG (0x0004)**

(reserved)																	UHCI_DMA_INFO_FULL_WM_INT_RAW																
																	UHCI_SEND_A_REG_Q_INT_RAW																
																	UHCI_SEND_S_REG_Q_INT_RAW																
																	UHCI_OUT_TOTAL_EOF_INT_RAW																
																	UHCI_OUTLINK_EOF_INT_RAW																
																	UHCI_IN_DSCR_EMPTY_INT_RAW																
																	UHCI_OUT_DSCR_ERR_INT_RAW																
																	UHCI_OUT_EOF_INT_RAW																
																	UHCI_IN_DONE_INT_RAW																
																	UHCI_IN_SUC_EOF_INT_RAW																
																	UHCI_TX_HUNG_INT_RAW																
																	UHCI_RX_HUNG_INT_RAW																
																	UHCI_TX_START_INT_RAW																
																	UHCI_RX_START_INT_RAW																

31																17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset							

**UHCI\_RX\_START\_INT\_RAW** This is the interrupt raw bit. Triggered when a separator char has been sent. (RO)

**UHCI\_TX\_START\_INT\_RAW** This is the interrupt raw bit. Triggered when DMA detects a separator char. (RO)

**UHCI\_RX\_HUNG\_INT\_RAW** This is the interrupt raw bit. Triggered when DMA takes more time to receive data than the configure value. (RO)

**UHCI\_TX\_HUNG\_INT\_RAW** This is the interrupt raw bit. Triggered when DMA takes more time to read data from RAM than the configured value. (RO)

**UHCI\_IN\_DONE\_INT\_RAW** This is the interrupt raw bit. Triggered when an receive descriptor is completed. (RO)

**UHCI\_IN\_SUC\_EOF\_INT\_RAW** This is the interrupt raw bit. Triggered when a data packet has been received successfully. (RO)

**UHCI\_IN\_ERR\_EOF\_INT\_RAW** This is the interrupt raw bit. Triggered when there are some errors in EOF in the receive descriptor. (RO)

**UHCI\_OUT\_DONE\_INT\_RAW** This is the interrupt raw bit. Triggered when an transmit descriptor is completed. (RO)

**UHCI\_OUT\_EOF\_INT\_RAW** This is the interrupt raw bit. Triggered when the current descriptor's EOF bit is 1. (RO)

**UHCI\_IN\_DSCR\_ERR\_INT\_RAW** This is the interrupt raw bit. Triggered when there are some errors in the receive descriptor. (RO)

**UHCI\_OUT\_DSCR\_ERR\_INT\_RAW** This is the interrupt raw bit. Triggered when there are some errors in the transmit descriptor. (RO)

**UHCI\_IN\_DSCR\_EMPTY\_INT\_RAW** This is the interrupt raw bit. Triggered when there are not enough inlinks for DMA. (RO)

**UHCI\_OUTLINK\_EOF\_ERR\_INT\_RAW** This is the interrupt raw bit. Triggered when there are some errors in EOF in the transmit descriptor. (RO)

Continued on the next page...

**Register 8.56: UHCI\_INT\_RAW\_REG (0x0004)**

Continued from the previous page...

**UHCI\_OUT\_TOTAL\_EOF\_INT\_RAW** This is the interrupt raw bit. Triggered when all data in the last buffer address has been sent out. (RO)

**UHCI\_SEND\_S\_REG\_Q\_INT\_RAW** This is the interrupt raw bit. Triggered when DMA has sent out a short packet using single\_send registers. (RO)

**UHCI\_SEND\_A\_REG\_Q\_INT\_RAW** This is the interrupt raw bit. Triggered when DMA has sent out a short packet using always\_send registers. (RO)

**UHCI\_DMA\_INFIFO\_FULL\_WM\_INT\_RAW** This is the interrupt raw bit. Triggered when the DMA INFIFO count has reached the configured threshold value. (RO)

**Register 8.57: UHCI\_INT\_ST\_REG (0x0008)**

(reserved)																	UHCI_DMA_INFO_FULL_WM_INT_ST UHCI_SEND_A_REG_Q_INT_ST UHCI_SEND_S_REG_Q_INT_ST UHCI_OUT_TOTAL_EOF_INT_ST UHCI_OUTLINK_EOF_ERR_INT_ST UHCI_IN_DSCR_EMPTY_INT_ST UHCI_OUT_DSCR_ERR_INT_ST UHCI_OUT_EOF_DONE_INT_ST UHCI_IN_ERR_EOF_INT_ST UHCI_IN_SUC_EOF_INT_ST UHCI_TX_DONE_INT_ST UHCI_RX_HUNG_INT_ST UHCI_TX_HUNG_INT_ST UHCI_RX_START_INT_ST UHCI_RX_START_INT_ST																			
31																	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset								

**UHCI\_RX\_START\_INT\_ST** This is the masked interrupt bit for UHCI\_RX\_START\_INT interrupt when UHCI\_RX\_START\_INT\_ENA is set to 1. (RO)

**UHCI\_TX\_START\_INT\_ST** This is the masked interrupt bit for UHCI\_TX\_START\_INT interrupt when UHCI\_TX\_START\_INT\_ENA is set to 1. (RO)

**UHCI\_RX\_HUNG\_INT\_ST** This is the masked interrupt bit for UHCI\_RX\_HUNG\_INT interrupt when UHCI\_RX\_HUNG\_INT\_ENA is set to 1. (RO)

**UHCI\_TX\_HUNG\_INT\_ST** This is the masked interrupt bit for UHCI\_TX\_HUNG\_INT interrupt when UHCI\_TX\_HUNG\_INT\_ENA is set to 1. (RO)

**UHCI\_IN\_DONE\_INT\_ST** This is the masked interrupt bit for UHCI\_IN\_DONE\_INT interrupt when UHCI\_IN\_DONE\_INT\_ENA is set to 1. (RO)

**UHCI\_IN\_SUC\_EOF\_INT\_ST** This is the masked interrupt bit for UHCI\_IN\_SUC\_EOF\_INT interrupt when UHCI\_IN\_SUC\_EOF\_INT\_ENA is set to 1. (RO)

**UHCI\_IN\_ERR\_EOF\_INT\_ST** This is the masked interrupt bit for UHCI\_IN\_ERR\_EOF\_INT interrupt when UHCI\_IN\_ERR\_EOF\_INT\_ENA is set to 1. (RO)

**UHCI\_OUT\_DONE\_INT\_ST** This is the masked interrupt bit for UHCI\_OUT\_DONE\_INT interrupt when UHCI\_OUT\_DONE\_INT\_ENA is set to 1. (RO)

**UHCI\_OUT\_EOF\_INT\_ST** This is the masked interrupt bit for UHCI\_OUT\_EOF\_INT interrupt when UHCI\_OUT\_EOF\_INT\_ENA is set to 1. (RO)

**UHCI\_IN\_DSCR\_ERR\_INT\_ST** This is the masked interrupt bit for UHCI\_IN\_DSCR\_ERR\_INT interrupt when UHCI\_IN\_DSCR\_ERR\_INT is set to 1. (RO)

**UHCI\_OUT\_DSCR\_ERR\_INT\_ST** This is the masked interrupt bit for UHCI\_OUT\_DSCR\_ERR\_INT interrupt when UHCI\_OUT\_DSCR\_ERR\_INT\_ENA is set to 1. (RO)

**UHCI\_IN\_DSCR\_EMPTY\_INT\_ST** This is the masked interrupt bit for UHCI\_IN\_DSCR\_EMPTY\_INT interrupt when UHCI\_IN\_DSCR\_EMPTY\_INT\_ENA is set to 1. (RO)

**UHCI\_OUTLINK\_EOF\_ERR\_INT\_ST** This is the masked interrupt bit for UHCI\_OUTLINK\_EOF\_ERR\_INT interrupt when UHCI\_OUTLINK\_EOF\_ERR\_INT\_ENA is set to 1. (RO)

Continued on the next page...

**Register 8.57: UHCI\_INT\_ST\_REG (0x0008)**

Continued from the previous page...

**UHCI\_OUT\_TOTAL\_EOF\_INT\_ST** This is the masked interrupt bit for UHCI\_OUT\_TOTAL\_EOF\_INT interrupt when UHCI\_OUT\_TOTAL\_EOF\_INT\_ENA is set to 1. (RO)

**UHCI\_SEND\_S\_REG\_Q\_INT\_ST** This is the masked interrupt bit for UHCI\_SEND\_S\_REG\_Q\_INT interrupt when UHCI\_SEND\_S\_REG\_Q\_INT\_ENA is set to 1. (RO)

**UHCI\_SEND\_A\_REG\_Q\_INT\_ST** This is the masked interrupt bit for UHCI\_SEND\_A\_REG\_Q\_INT interrupt when UHCI\_SEND\_A\_REG\_Q\_INT\_ENA is set to 1. (RO)

**UHCI\_DMA\_INFIFO\_FULL\_WM\_INT\_ST** This is the masked interrupt bit for UHCI\_DMA\_INFIFO\_FULL\_WM\_INT INTERRUPT when UHCI\_DMA\_INFIFO\_FULL\_WM\_INT\_ENA is set to 1. (RO)



Register 8.58: UHCI\_INT\_ENA\_REG (0x000C)

(reserved)																																UHCI_DMA_INFO_FULL_WM_INT_ENA UHCI_SEND_A_REG_Q_INT_ENA UHCI_SEND_S_REG_Q_INT_ENA UHCI_OUT_TOTAL_EOF_INT_ENA UHCI_OUTLINK_EOF_INT_ENA UHCI_IN_DSCR_EMPTY_INT_ENA UHCI_OUT_DSCR_ERR_INT_ENA UHCI_OUT_EOF_INT_ENA UHCI_IN_DSCR_EMPTY_INT_ENA UHCI_OUT_DONE_INT_ENA UHCI_IN_SUC_EOF_INT_ENA UHCI_TX_HUNG_INT_ENA UHCI_RX_HUNG_INT_ENA UHCI_TX_START_INT_ENA UHCI_RX_START_INT_ENA																		
31																17																16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0																0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**UHCI\_RX\_START\_INT\_ENA** This is the interrupt enable bit for UHCI\_RX\_START\_INT interrupt. (R/W)

**UHCI\_TX\_START\_INT\_ENA** This is the interrupt enable bit for UHCI\_TX\_START\_INT interrupt. (R/W)

**UHCI\_RX\_HUNG\_INT\_ENA** This is the interrupt enable bit for UHCI\_RX\_HUNG\_INT interrupt. (R/W)

**UHCI\_TX\_HUNG\_INT\_ENA** This is the interrupt enable bit for UHCI\_TX\_HUNG\_INT interrupt. (R/W)

**UHCI\_IN\_DONE\_INT\_ENA** This is the interrupt enable bit for UHCI\_IN\_DONE\_INT interrupt. (R/W)

**UHCI\_IN\_SUC\_EOF\_INT\_ENA** This is the interrupt enable bit for UHCI\_IN\_SUC\_EOF\_INT interrupt. (R/W)

**UHCI\_IN\_ERR\_EOF\_INT\_ENA** This is the interrupt enable bit for UHCI\_IN\_ERR\_EOF\_INT interrupt. (R/W)

**UHCI\_OUT\_DONE\_INT\_ENA** This is the interrupt enable bit for UHCI\_OUT\_DONE\_INT interrupt. (R/W)

**UHCI\_OUT\_EOF\_INT\_ENA** This is the interrupt enable bit for UHCI\_OUT\_EOF\_INT interrupt. (R/W)

**UHCI\_IN\_DSCR\_ERR\_INT\_ENA** This is the interrupt enable bit for UHCI\_IN\_DSCR\_ERR\_INT interrupt. (R/W)

**UHCI\_OUT\_DSCR\_ERR\_INT\_ENA** This is the interrupt enable bit for UHCI\_OUT\_DSCR\_ERR\_INT interrupt. (R/W)

**UHCI\_IN\_DSCR\_EMPTY\_INT\_ENA** This is the interrupt enable bit for UHCI\_IN\_DSCR\_EMPTY\_INT interrupt. (R/W)

**UHCI\_OUTLINK\_EOF\_ERR\_INT\_ENA** This is the interrupt enable bit for UHCI\_OUTLINK\_EOF\_ERR\_INT interrupt. (R/W)

**UHCI\_OUT\_TOTAL\_EOF\_INT\_ENA** This is the interrupt enable bit for UHCI\_OUT\_TOTAL\_EOF\_INT interrupt. (R/W)

**UHCI\_SEND\_S\_REG\_Q\_INT\_ENA** This is the interrupt enable bit for UHCI\_SEND\_S\_REG\_Q\_INT interrupt. (R/W)

Continued on the next page...

**Register 8.58: UHCI\_INT\_ENA\_REG (0x000C)**

Continued from the previous page...

**UHCI\_SEND\_A\_REG\_Q\_INT\_ENA** This is the interrupt enable bit for UHCI\_SEND\_A\_REG\_Q\_INT interrupt. (R/W)

**UHCI\_DMA\_INFIFO\_FULL\_WM\_INT\_ENA** This is the interrupt enable bit for UHCI\_DMA\_INFIFO\_FULL\_WM\_INT interrupt. (R/W)

**Register 8.59: UHCI\_INT\_CLR\_REG (0x0010)**

(reserved)																UHCI_DMA_INFIFO_FULL_WM_INT_CLR UHCI_SEND_A_REG_Q_INT_CLR UHCI_SEND_S_REG_Q_INT_CLR UHCI_OUT_TOTAL_EOF_INT_CLR UHCI_OUTLINK_EOF_ERR_INT_CLR UHCI_IN_DSCR_EMPTY_INT_CLR UHCI_OUT_DSCR_ERR_INT_CLR UHCI_OUT_EOF_INT_CLR UHCI_IN_ERR_EOF_INT_CLR UHCI_IN_SUC_EOF_INT_CLR UHCI_TX_HUNG_INT_CLR UHCI_RX_HUNG_INT_CLR UHCI_TX_START_INT_CLR UHCI_RX_START_INT_CLR																	
31																17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset							

**UHCI\_RX\_START\_INT\_CLR** Set this bit to clear UHCI\_RX\_START\_INT interrupt. (WO)

**UHCI\_TX\_START\_INT\_CLR** Set this bit to clear UHCI\_TX\_START\_INT interrupt. (WO)

**UHCI\_RX\_HUNG\_INT\_CLR** Set this bit to clear UHCI\_RX\_HUNG\_INT interrupt. (WO)

**UHCI\_TX\_HUNG\_INT\_CLR** Set this bit to clear UHCI\_TX\_HUNG\_INT interrupt. (WO)

**UHCI\_IN\_DONE\_INT\_CLR** Set this bit to clear UHCI\_IN\_DONE\_INT interrupt. (WO)

**UHCI\_IN\_SUC\_EOF\_INT\_CLR** Set this bit to clear UHCI\_IN\_SUC\_EOF\_INT interrupt. (WO)

**UHCI\_IN\_ERR\_EOF\_INT\_CLR** Set this bit to clear UHCI\_IN\_ERR\_EOF\_INT interrupt. (WO)

**UHCI\_OUT\_DONE\_INT\_CLR** Set this bit to clear UHCI\_OUT\_DONE\_INT interrupt. (WO)

**UHCI\_OUT\_EOF\_INT\_CLR** Set this bit to clear UHCI\_OUT\_EOF\_INT interrupt. (WO)

**UHCI\_IN\_DSCR\_ERR\_INT\_CLR** Set this bit to clear UHCI\_IN\_DSCR\_ERR\_INT interrupt. (WO)

**UHCI\_OUT\_DSCR\_ERR\_INT\_CLR** Set this bit to clear UHCI\_OUT\_DSCR\_ERR\_INT interrupt. (WO)

**UHCI\_IN\_DSCR\_EMPTY\_INT\_CLR** Set this bit to clear UHCI\_IN\_DSCR\_EMPTY\_INT interrupt. (WO)

**UHCI\_OUTLINK\_EOF\_ERR\_INT\_CLR** Set this bit to clear UHCI\_OUTLINK\_EOF\_ERR\_INT interrupt. (WO)

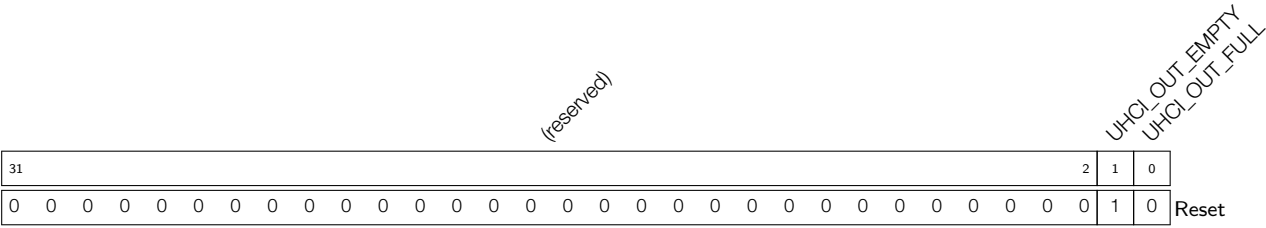
**UHCI\_OUT\_TOTAL\_EOF\_INT\_CLR** Set this bit to clear UHCI\_OUT\_TOTAL\_EOF\_INT interrupt. (WO)

**UHCI\_SEND\_S\_REG\_Q\_INT\_CLR** Set this bit to clear UHCI\_SEND\_S\_UHCI\_Q\_INT interrupt. (WO)

**UHCI\_SEND\_A\_REG\_Q\_INT\_CLR** Set this bit to clear UHCI\_SEND\_A\_UHCI\_Q\_INT interrupt. (WO)

**UHCI\_DMA\_INFIFO\_FULL\_WM\_INT\_CLR** Set this bit to clear UHCI\_DMA\_INFIFO\_FULL\_WM\_INT interrupt. (WO)

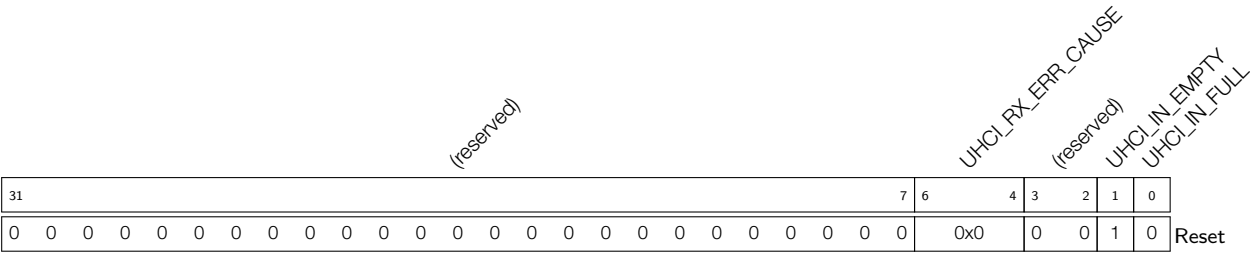
Register 8.60: UHCI\_DMA\_OUT\_STATUS\_REG (0x0014)



**UHCI\_OUT\_FULL** 1: DMA data-output FIFO is full. (RO)

**UHCI\_OUT\_EMPTY** 1: DMA data-output FIFO is empty. (RO)

Register 8.61: UHCI\_DMA\_IN\_STATUS\_REG (0x001C)



**UHCI\_IN\_FULL** Data-input FIFO full signal. (RO)

**UHCI\_IN\_EMPTY** Data-input FIFO empty signal. (RO)

**UHCI\_RX\_ERR\_CAUSE** This register indicates the error type when DMA has received a packet with error. 3'b001: Checksum error in HCI packet; 3'b010: Sequence number error in HCI packet; 3'b011: CRC bit error in HCI packet; 3'b100: 0xc0 is found but received HCI packet is not end; 3'b101: 0xc0 is not found when receiving HCI packet is end; 3'b110: CRC check error. (RO)

**Register 8.62: UHCI\_STATE0\_REG (0x0030)**

(reserved)															UHCI_DECODE_STATE															UHCI_INFIFO_CNT_DEBUG															UHCI_IN_STATE															UHCI_IN_DSCR_STATE															UHCI_INLINK_DSCR_ADDR														
31	30	28		27	23			22	20		19	18	17																		0																																																										
0	0		0			0			0	0	0																	Reset																																																													

Reset

**UHCI\_INLINK\_DSCR\_ADDR** This register stores the current receive descriptor's address. (RO)

**UHCI\_IN\_DSCR\_STATE** Reserved. (RO)

**UHCI\_IN\_STATE** Reserved. (RO)

**UHCI\_INFIFO\_CNT\_DEBUG** This register stores the byte number of the data in the receive descriptor's FIFO. (RO)

**UHCI\_DECODE\_STATE** UHCI decoder status. (RO)

**Register 8.63: UHCI\_STATE1\_REG (0x0034)**

(reserved)																	UHCI_ENCODE_STATE											UHCI_OUTFIFO_CNT											UHCI_OUT_STATE											UHCI_OUT_DSCR_STATE											UHCI_OUTLINK_DSCR_ADDR																														
31	30	28		27	23			22	20		19	18	17	0																																																																													
0	0		0			0		0	0		0																															Reset																																																	

Reset

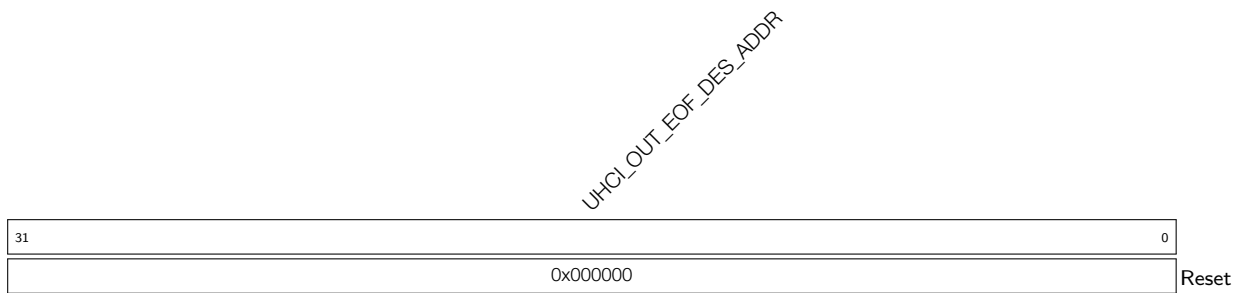
**UHCI\_OUTLINK\_DSCR\_ADDR** This register stores the current transmit descriptor's address. (RO)

**UHCI\_OUT\_DSCR\_STATE** Reserved. (RO)

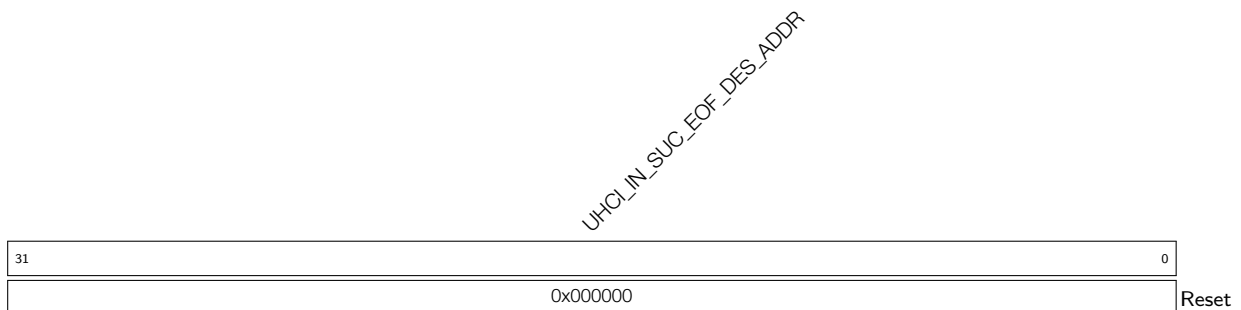
**UHCI\_OUT\_STATE** Reserved. (RO)

**UHCI\_OUTFIFO\_CNT** This register stores the byte number of the data in the transmit descriptor's FIFO. (RO)

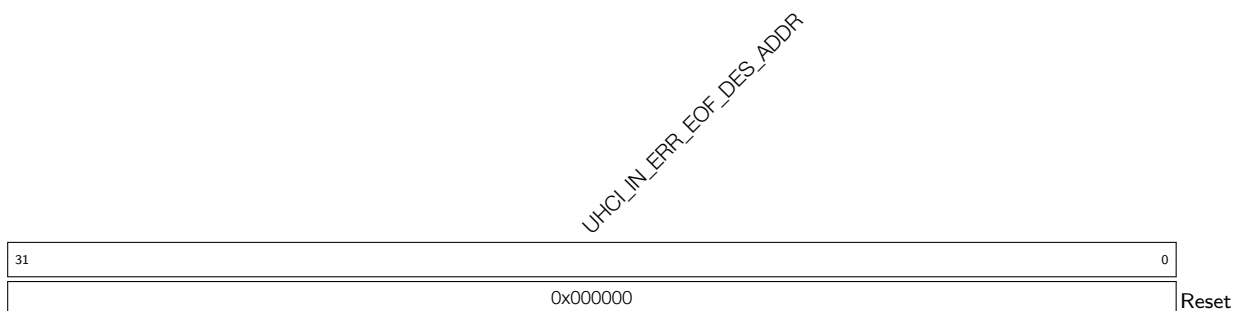
**UHCI\_ENCODE\_STATE** UHCI encoder status. (RO)

**Register 8.64: UHCI\_DMA\_OUT\_EOF\_DES\_ADDR\_REG (0x0038)**

**UHCI\_OUT\_EOF\_DES\_ADDR** This register stores the address of the transmit descriptor when the EOF bit in this descriptor is 1. (RO)

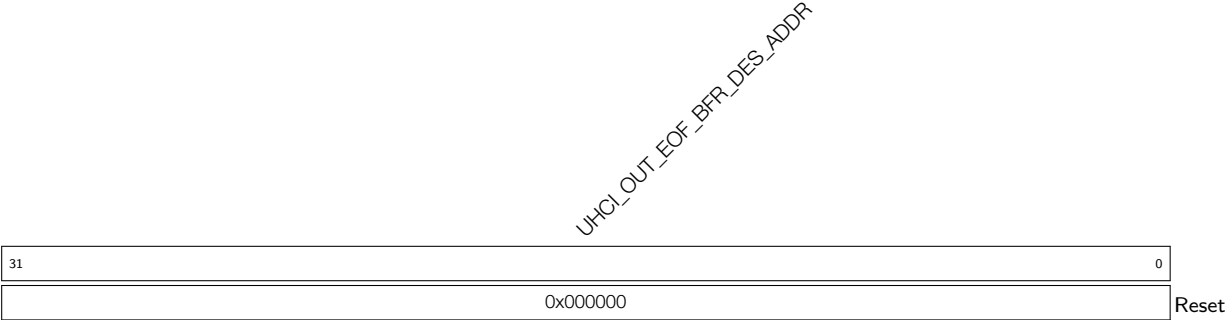
**Register 8.65: UHCI\_DMA\_IN\_SUC\_EOF\_DES\_ADDR\_REG (0x003C)**

**UHCI\_IN\_SUC\_EOF\_DES\_ADDR** This register stores the address of the receive descriptor when received successful EOF. (RO)

**Register 8.66: UHCI\_DMA\_IN\_ERR\_EOF\_DES\_ADDR\_REG (0x0040)**

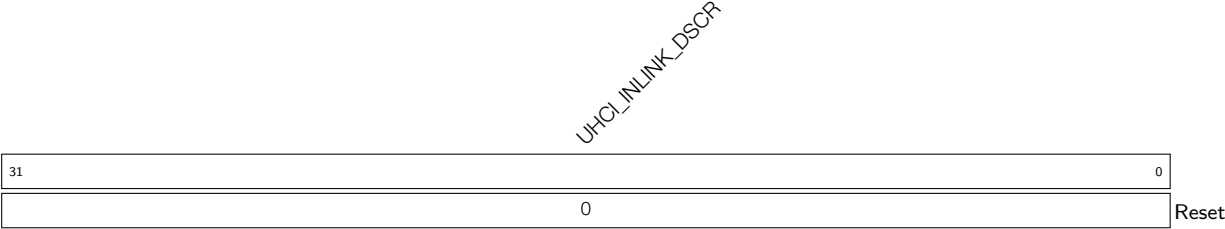
**UHCI\_IN\_ERR\_EOF\_DES\_ADDR** This register stores the address of the receive descriptor when there are some errors in this descriptor. (RO)

Register 8.67: UHCI\_DMA\_OUT\_EOF\_BFR\_DES\_ADDR\_REG (0x0044)



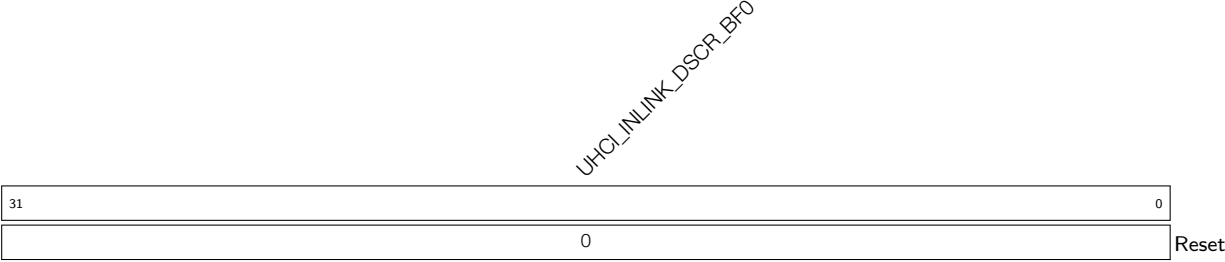
**UHCI\_OUT\_EOF\_BFR\_DES\_ADDR** This register stores the address of the transmit descriptor before the last transmit descriptor. (RO)

Register 8.68: UHCI\_DMA\_IN\_DSCR\_REG (0x004C)



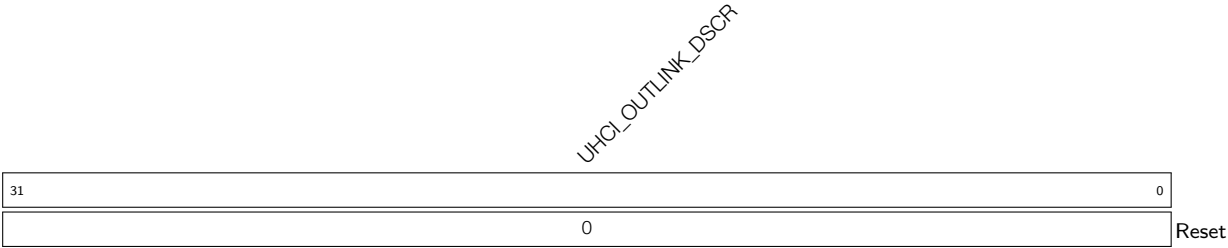
**UHCI\_INLINK\_DSCR** This register stores the third word of the next receive descriptor. (RO)

Register 8.69: UHCI\_DMA\_IN\_DSCR\_BF0\_REG (0x0050)



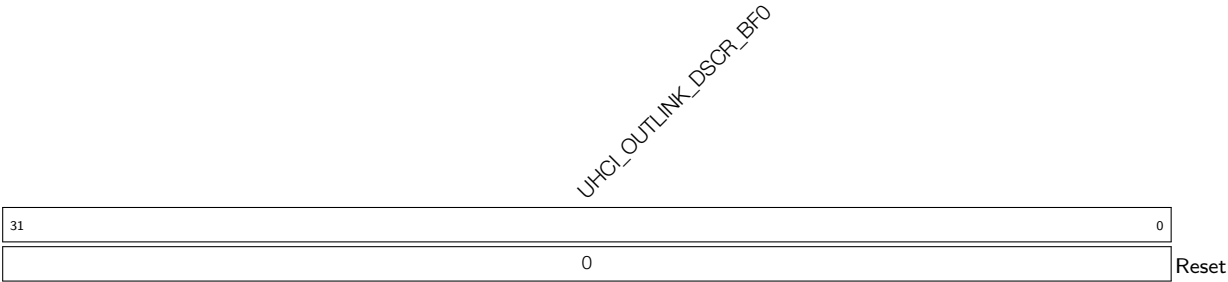
**UHCI\_INLINK\_DSCR\_BF0** This register stores the third word of the current receive descriptor. (RO)

Register 8.70: UHCI\_DMA\_OUT\_DSCR\_REG (0x0058)



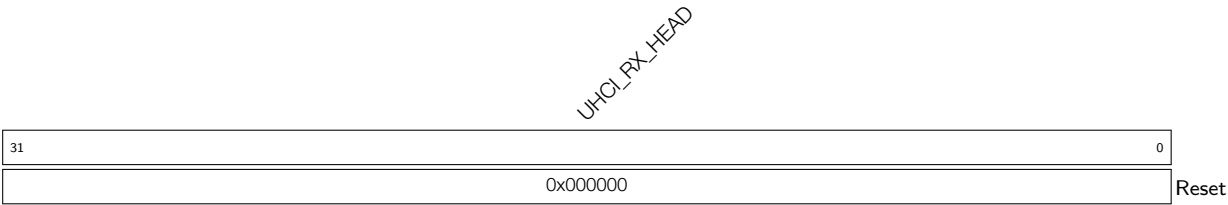
**UHCI\_OUTLINK\_DSCR** This register stores the third word of the next transmit descriptor. (RO)

Register 8.71: UHCI\_DMA\_OUT\_DSCR\_BF0\_REG (0x005C)



**UHCI\_OUTLINK\_DSCR\_BF0** This register stores the third word of the current transmit descriptor. (RO)

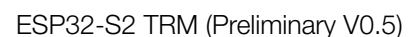
Register 8.72: UHCI\_RX\_HEAD\_REG (0x0070)



**UHCI\_RX\_HEAD** This register stores the header of the current received packet. (RO)



## 265



**UHCI\_OUTFIFO\_PUSH** Set this bit to push data into the data-output FIFO. (R/W)

Register 8.76: UHCI\_DMA\_IN\_LINK\_REG (0x0028)

UHCI_INLINK_PARK				UHCI_INLINK_RESTART				UHCI_INLINK_START				UHCI_INLINK_STOP				(reserved)				UHCI_INLINK_AUTO_RET				UHCI_INLINK_ADDR												
31	30	29	28	27											21	20	19											0								
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0x000														Reset								

**UHCI\_INLINK\_ADDR** This register is used to specify the least significant 20 bits of the first receive descriptor's address. (R/W)

**UHCI\_INLINK\_AUTO\_RET** This is the enable bit to return to current receive descriptor's address, when there are some errors in current packet. (R/W)

**UHCI\_INLINK\_STOP** Set this bit to stop dealing with the receive descriptors. (R/W)

**UHCI\_INLINK\_START** Set this bit to start dealing with the receive descriptors. (R/W)

**UHCI\_INLINK\_RESTART** Set this bit to restart new receive descriptors. (R/W)

**UHCI\_INLINK\_PARK** 1: the receive descriptor's FSM is in idle state. 0: the receive descriptor's FSM is working. (RO)

Register 8.77: UHCI\_DATE\_REG (0x00FC)

UHCI_DATE																															
31																															0
0x18073001																															
Reset																															

**UHCI\_DATE** This is the version control register. (R/W)

## 9. LED PWM Controller

### 9.1 Overview

The LED PWM controller is primarily designed to control LED devices, as well as generate PWM signals. It has 14-bit timers and waveform generators.

### 9.2 Features

The LED PWM controller has the following features:

- Four independent timers that support division by fractions
- Eight independent waveform generators able to produce eight PWM signals
- Fading duty cycle of PWM signals without interference from any processors. An interrupt can be generated after the fade has completed
- Adjustable phase of PWM signal output
- PWM signal output in low-power mode

For the convenience of description, in the following sections the eight waveform generators are collectively referred to as PWM $n$ , and the four timers are collectively referred to as Timer $x$ .

### 9.3 Functional Description

#### 9.3.1 Architecture

Figure 9-1 shows the architecture of the LED PWM controller. Figure 9-2 illustrates a PWM generator with its selected timer and a counter.

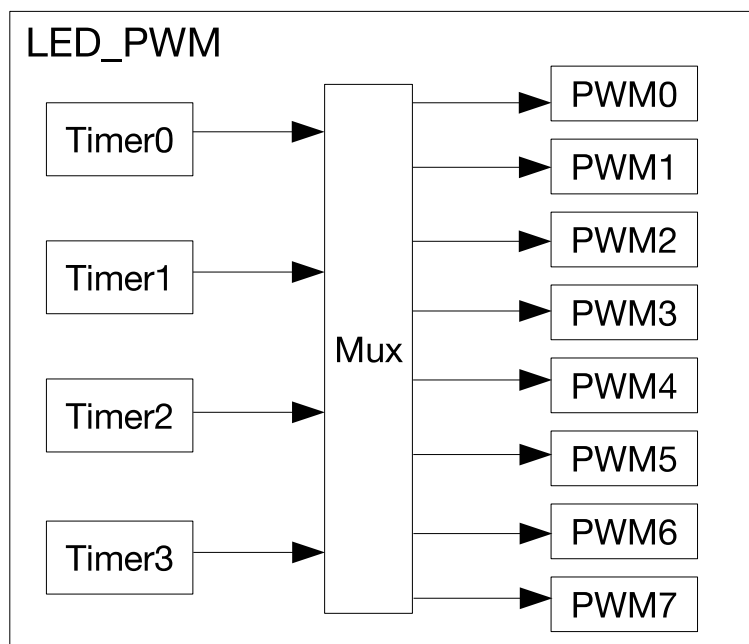


Figure 9-1. LED\_PWM Architecture

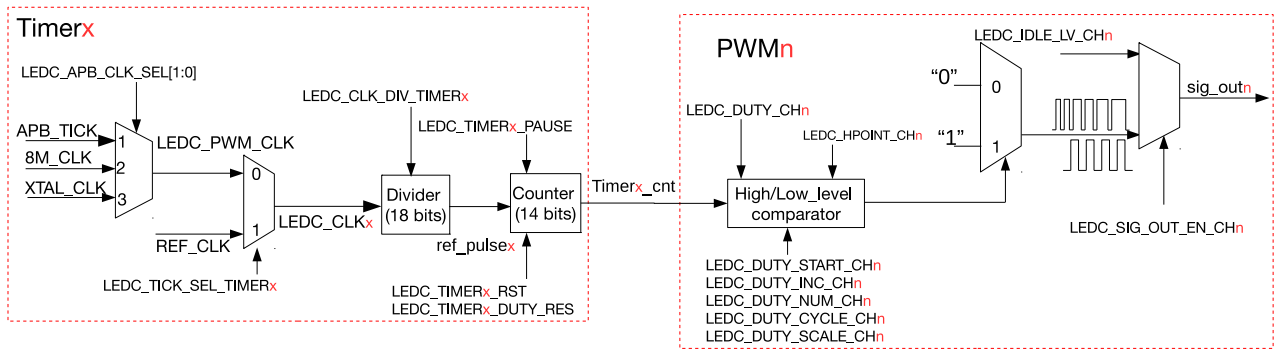


Figure 9-2. LED\_PWM generator Diagram

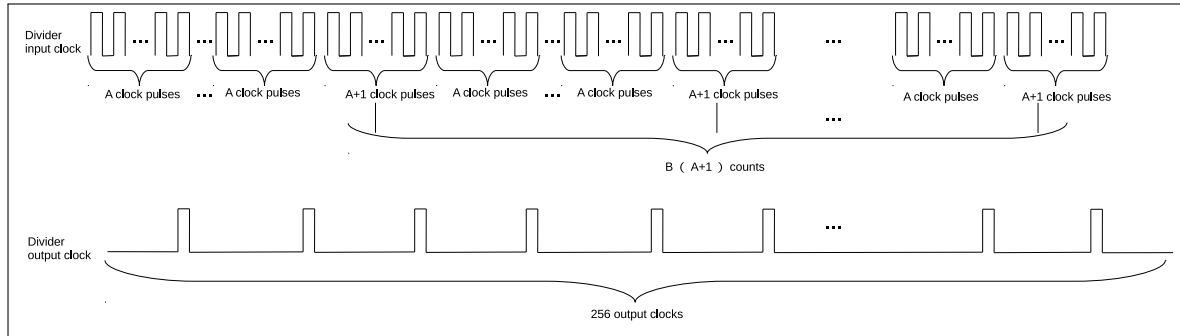


Figure 9-3. LED\_PWM Divider

### 9.3.2 Timers

The clock of the LED PWM controller, LEDC\_PWM\_CLK, has three clock sources: APB\_CLK, RTC8M\_CLK and XTAL\_CLK, selected by configuring LEDC\_APB\_CLK\_SEL[1:0]. The clock of each LED PWM timer, LEDC\_CLK<sub>x</sub>, has two clock sources: LEDC\_PWM\_CLK or REF\_TICK. When REF\_TICK is used as the clock source of a timer, LEDC\_APB\_CLK\_SEL[1:0] should be set 1 and the cycle of REF\_TICK should be an integral multiple of APB\_CLK cycles. Otherwise this clock will be not accurate. For more information on the clock sources, please see Chapter [Reset and Clock](#).

The output clock derived from LEDC\_CLK<sub>x</sub> is used as the base clock for the counter. The divider's divisor is configured by LEDC\_CLK\_DIV\_TIMER<sub>x</sub>. It is a fixed-point number: the highest 10 bits is the integer part represented as A, while the lowest eight bits is the fractional part represented as B. This divisor LEDC\_CLK\_DIV<sub>x</sub> is calculated as:

$$LEDC\_CLK\_DIV_x = A + \frac{B}{256}$$

When the fractional part B is not 0, the input and output clock of the divider is shown as in figure 9-3. Among the 256 output clocks, B of them are divided by (A+1), whereas the remaining (256-B) are divided by A. Output clocks divided by (A+1) are evenly distributed in the total 256 output clocks.

The LED PWM controller has a 14-bit counter that counts up to  $2^{LEDC\_TIMER_x\_DUTY\_RES} - 1$ . If the counting value reaches  $2^{LEDC\_TIMER_x\_DUTY\_RES} - 1$ , the counter will overflow and restart counting from 0. The counting value can be reset, suspended or read by software. A LEDC\_TIMER<sub>x</sub>\_OVF\_INT interrupt can be generated every time when the counter overflows or when it overflows for (LEDC\_OVF\_NUM\_CH<sub>n</sub> + 1) times. The interrupt configuration is as follows:

1. Set LEDC\_OVF\_CNT\_EN\_CH<sub>n</sub>
2. Configure LEDC\_OVF\_NUM\_CH<sub>n</sub> with the times of overflow minus 1

3. Set `LEDC_OVF_CNT_CH $n$ _INT_ENA`

4. Set `LEDC_TIMER $x$ _DUTY_RES` to enable the timer and wait for a `LEDC_OVF_CNT_CH $n$ _INT` interrupt

The frequency of a PWM generator output signal, `sig_out $n$` , depends on both the divisor of the divider, as well as the range of the counter:

$$f_{\text{sig\_out}n} = \frac{f_{\text{LEDC\_CLK}x}}{\text{LEDC\_CLK\_DIV}x \cdot 2^{\text{LEDC\_TIMER}x\_DUTY\_RES}}$$

To change the divisor and times of overflow, you should configure `LEDC_CLK_DIV_TIMER $x$`  and `LEDC_TIMER $x$ _DUTY_RES` respectively, and then set `LEDC_TIMER $x$ _PARA_UP`; otherwise this change is not valid. The newly configured value is updated upon next overflow of the counter.

### 9.3.3 PWM Generators

As shown in figure 9-2, each PWM generator has a high/low level comparator and two selectors. A PWM generator takes the 14-bit counting value of the selected timer, compares it to values of the comparator `Hpoint $n$`  and `Lpoint $n$` , and therefore control the level of PWM signals.

- If `Timer $x$ _cnt == Hpoint $n$` , `sig_out $n$`  is 1.
- If `Timer $x$ _cnt == Lpoint $n$` , `sig_out $n$`  is 0.

`Hpoint $n$`  is updated by `LEDC_HPOINT_CH $n$`  when the counter overflows. The initial value of `Lpoint $n$`  is the sum of `LEDC_DUTY_CH $n$ [18..4]` and `LEDC_HPOINT_CH $n$`  when the counter overflows. By configuring these two fields, the relative phase and the duty cycle of the PWM output can be set.

Figure 9-4 illustrates PWM's waveform when the duty cycle is fixed.

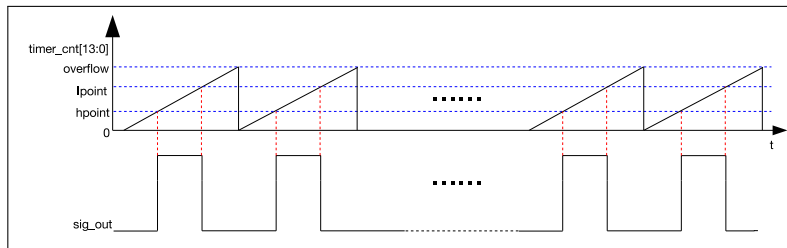


Figure 9-4. LED\_PWM Output Signal Diagram

`LEDC_DUTY_CH $n$`  is a fixed-point register with four fractional bits. `LEDC_DUTY_CH $n$ [18..4]` is the integral part used directly for PWM calculation. `LEDC_DUTY_CH $n$ [3..0]` is the fractional part used to dither the output. If `LEDC_DUTY_CH $n$ [3..0]` is non-zero, then among every 16 cycles of `sig_out $n$` , `LEDC_DUTY_CH $n$ [3..0]` have PWM pulses with width one timer cycle longer than that of `(16-LEDC_DUTY_CH $n$ [3..0])`. This feature effectively increases the resolution of the PWM generator to 18 bits.

### 9.3.4 Duty Cycle Fading

The PWM generators is able to fade the duty cycle, that is to gradually change the duty cycle from one value to another. This is achieved by configuring `LEDC_DUTY_CH $n$` , `LEDC_DUTY_START_CH $n$` , `LEDC_DUTY_INC_CH $n$` , `LEDC_DUTY_NUM_CH $n$`  and `LEDC_DUTY_SCALE_CH $n$` .

`LEDC_DUTY_START_CH $n$`  is used to update the value of `Lpoint $n$` . When this bit is set and the counter overflows, `Lpoint $n$`  increments or decrements automatically, depending on whether the bit `LEDC_DUTY_INC_CH $n$`  is set or cleared.

The duty cycle of sig\_out $n$  changes every LEDC\_DUTY\_CYCLE\_CH $n$  PWM pulse cycles by adding or subtracting the value of LEDC\_DUTY\_SCALE\_CH $n$ .

Figure 9-5 is a diagram of fading duty cycle. Upon reaching LEDC\_DUTY\_NUM\_CH $n$ , the fade stops and

LEDC\_DUTY\_CHNG\_END\_CH $n$ \_INT interrupt is generated. When configured like this, the duty cycle of sig\_out $n$  increases by LEDC\_DUTY\_SCALE\_CH $n$  every LEDC\_DUTY\_CYCLE\_CH $n$  PWM pulse cycles.

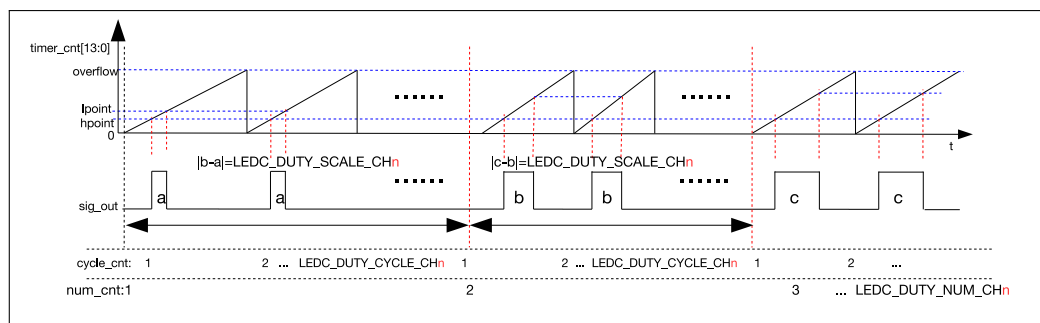


Figure 9-5. Output Signal Diagram of Fading Duty Cycle

LEDC\_SIG\_OUT\_EN\_CH $n$  used to enable PWM waveform output. When LEDC\_SIG\_OUT\_EN\_CH $n$  is 0, the level of sig\_out $n$  is constant as specified in LEDC\_IDLE\_LV\_CH $n$ .

If LEDC\_HPOINT\_CH $n$ , LEDC\_DUTY\_START\_CH $n$ , LEDC\_SIG\_OUT\_EN\_CH $n$ , LEDC\_TIMER\_SEL\_CH $n$ , LEDC\_DUTY\_NUM\_CH $n$ , LEDC\_DUTY\_CYCLE\_CH $n$ , LEDC\_DUTY\_SCALE\_CH $n$ , LEDC\_DUTY\_INC\_CH $n$  and LEDC\_OVF\_CNT\_EN\_CH $n$  are reconfigured, LEDC\_PARA\_UP\_CH $n$  should be set to apply the new configuration.

### 9.3.5 Interrupts

- LEDC\_OVF\_CNT\_CH $n$ \_INT: Triggered when the timer counter overflows for (LEDC\_OVF\_NUM\_CH $n$  + 1) times and register LEDC\_OVF\_CNT\_EN\_CH $n$  is set to 1.
- LEDC\_DUTY\_CHNG\_END\_CH $n$ \_INT: Triggered when a fade on a LED PWM generator has finished.
- LEDC\_TIMER $x$ \_OVF\_INT: Triggered when a LED PWM timer has reached its maximum counter value.

## 9.4 Base Address

Users can access the LED PWM controller with two base addresses, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter 1 *System and Memory*.

Table 43: LED\_PWM Base Address

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F419000
PeriBUS2	0x60019000

## 9.5 Register Summary

The addresses in the following table are relative to the LED PWM controller base addresses provided in Section 9.4.

Name	Description	Address	Access
<b>Configuration Register</b>			
<a href="#">LEDC_CH0_CONF0_REG</a>	Configuration register 0 for channel 0	0x0000	varies
<a href="#">LEDC_CH0_CONF1_REG</a>	Configuration register 1 for channel 0	0x000C	R/W
<a href="#">LEDC_CH1_CONF0_REG</a>	Configuration register 0 for channel 1	0x0014	varies
<a href="#">LEDC_CH1_CONF1_REG</a>	Configuration register 1 for channel 1	0x0020	R/W
<a href="#">LEDC_CH2_CONF0_REG</a>	Configuration register 0 for channel 2	0x0028	varies
<a href="#">LEDC_CH2_CONF1_REG</a>	Configuration register 1 for channel 2	0x0034	R/W
<a href="#">LEDC_CH3_CONF0_REG</a>	Configuration register 0 for channel 3	0x003C	varies
<a href="#">LEDC_CH3_CONF1_REG</a>	Configuration register 1 for channel 3	0x0048	R/W
<a href="#">LEDC_CH4_CONF0_REG</a>	Configuration register 0 for channel 4	0x0050	varies
<a href="#">LEDC_CH4_CONF1_REG</a>	Configuration register 1 for channel 4	0x005C	R/W
<a href="#">LEDC_CH5_CONF0_REG</a>	Configuration register 0 for channel 5	0x0064	varies
<a href="#">LEDC_CH5_CONF1_REG</a>	Configuration register 1 for channel 5	0x0070	R/W
<a href="#">LEDC_CH6_CONF0_REG</a>	Configuration register 0 for channel 6	0x0078	varies
<a href="#">LEDC_CH6_CONF1_REG</a>	Configuration register 1 for channel 6	0x0084	R/W
<a href="#">LEDC_CH7_CONF0_REG</a>	Configuration register 0 for channel 7	0x008C	varies
<a href="#">LEDC_CH7_CONF1_REG</a>	Configuration register 1 for channel 7	0x0098	R/W
<a href="#">LEDC_CONF_REG</a>	Global ledc configuration register	0x00D0	R/W
<b>Hpoint Register</b>			
<a href="#">LEDC_CH0_HPOINT_REG</a>	High point register for channel 0	0x0004	R/W
<a href="#">LEDC_CH1_HPOINT_REG</a>	High point register for channel 1	0x0018	R/W
<a href="#">LEDC_CH2_HPOINT_REG</a>	High point register for channel 2	0x002C	R/W
<a href="#">LEDC_CH3_HPOINT_REG</a>	High point register for channel 3	0x0040	R/W
<a href="#">LEDC_CH4_HPOINT_REG</a>	High point register for channel 4	0x0054	R/W
<a href="#">LEDC_CH5_HPOINT_REG</a>	High point register for channel 5	0x0068	R/W
<a href="#">LEDC_CH6_HPOINT_REG</a>	High point register for channel 6	0x007C	R/W
<a href="#">LEDC_CH7_HPOINT_REG</a>	High point register for channel 7	0x0090	R/W
<b>Duty Cycle Register</b>			
<a href="#">LEDC_CH0_DUTY_REG</a>	Initial duty cycle for channel 0	0x0008	R/W
<a href="#">LEDC_CH0_DUTY_R_REG</a>	Current duty cycle for channel 0	0x0010	RO
<a href="#">LEDC_CH1_DUTY_REG</a>	Initial duty cycle for channel 1	0x001C	R/W
<a href="#">LEDC_CH1_DUTY_R_REG</a>	Current duty cycle for channel 1	0x0024	RO
<a href="#">LEDC_CH2_DUTY_REG</a>	Initial duty cycle for channel 2	0x0030	R/W
<a href="#">LEDC_CH2_DUTY_R_REG</a>	Current duty cycle for channel 2	0x0038	RO
<a href="#">LEDC_CH3_DUTY_REG</a>	Initial duty cycle for channel 3	0x0044	R/W
<a href="#">LEDC_CH3_DUTY_R_REG</a>	Current duty cycle for channel 3	0x004C	RO
<a href="#">LEDC_CH4_DUTY_REG</a>	Initial duty cycle for channel 4	0x0058	R/W
<a href="#">LEDC_CH4_DUTY_R_REG</a>	Current duty cycle for channel 4	0x0060	RO
<a href="#">LEDC_CH5_DUTY_REG</a>	Initial duty cycle for channel 5	0x006C	R/W
<a href="#">LEDC_CH5_DUTY_R_REG</a>	Current duty cycle for channel 5	0x0074	RO

Name	Description	Address	Access
<a href="#">LEDC_CH6_DUTY_REG</a>	Initial duty cycle for channel 6	0x0080	R/W
<a href="#">LEDC_CH6_DUTY_R_REG</a>	Current duty cycle for channel 6	0x0088	RO
<a href="#">LEDC_CH7_DUTY_REG</a>	Initial duty cycle for channel 7	0x0094	R/W
<a href="#">LEDC_CH7_DUTY_R_REG</a>	Current duty cycle for channel 7	0x009C	RO
<b>Timer Register</b>			
<a href="#">LEDC_TIMER0_CONF_REG</a>	Timer 0 configuration	0x00A0	varies
<a href="#">LEDC_TIMER0_VALUE_REG</a>	Timer 0 current counter value	0x00A4	RO
<a href="#">LEDC_TIMER1_CONF_REG</a>	Timer 1 configuration	0x00A8	varies
<a href="#">LEDC_TIMER1_VALUE_REG</a>	Timer 1 current counter value	0x00AC	RO
<a href="#">LEDC_TIMER2_CONF_REG</a>	Timer 2 configuration	0x00B0	varies
<a href="#">LEDC_TIMER2_VALUE_REG</a>	Timer 2 current counter value	0x00B4	RO
<a href="#">LEDC_TIMER3_CONF_REG</a>	Timer 3 configuration	0x00B8	varies
<a href="#">LEDC_TIMER3_VALUE_REG</a>	Timer 3 current counter value	0x00BC	RO
<b>Interrupt Register</b>			
<a href="#">LEDC_INT_RAW_REG</a>	Raw interrupt status	0x00C0	RO
<a href="#">LEDC_INT_ST_REG</a>	Masked interrupt status	0x00C4	RO
<a href="#">LEDC_INT_ENA_REG</a>	Interrupt enable bits	0x00C8	R/W
<a href="#">LEDC_INT_CLR_REG</a>	Interrupt clear bits	0x00CC	WO
<b>Version Register</b>			
<a href="#">LEDC_DATE_REG</a>	Version control register	0x00FC	R/W



## 9.6 Registers

Register 9.1: LEDC\_CH $n$ \_CONF0\_REG ( $n$ : 0-7) (0x0000+0x14\* $n$ )

(reserved)																LEDC_OVF_CNT_RESET_ST_CH <sup>n</sup>			LEDC_OVF_CNT_RESET_CH <sup>n</sup>			LEDC_OVF_CNT_EN_CH <sup>n</sup>			LEDC_OVF_NUM_CH <sup>n</sup>				LEDC_PARA_UP_CH <sup>n</sup>		LEDC_IDLE_LV_CH <sup>n</sup>		LEDC_SIG_OUT_EN_CH <sup>n</sup>		LEDC_TIMER_SEL_CH <sup>n</sup>	
31																18		17	16	15	14				5				4	3	2	1	0			
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0	0	0	0x0				0				0	0	0x0		Reset					

**LEDC\_TIMER\_SEL\_CH $n$**  This field is used to select one of timers for channel  $n$ .

0: select timer0

1: select timer1

2: select timer2

3: select timer3 (R/W)

**LEDC\_SIG\_OUT\_EN\_CH $n$**  Set this bit to enable signal output on channel  $n$ . (R/W)

**LEDC\_IDLE\_LV\_CH $n$**  This bit is used to control the output value when channel  $n$  is inactive. (R/W)

**LEDC\_PARA\_UP\_CH $n$**  This bit is used to update register LEDC\_CH $n$ \_HPOINT and LEDC\_CH $n$ \_DUTY for channel  $n$ . (WO)

**LEDC\_OVF\_NUM\_CH $n$**  This register is used to configure the maximum times of overflow minus 1.

The LEDC\_OVF\_CNT\_CH $n$ \_INT interrupt will be triggered when channel  $n$  overflows for (LEDC\_OVF\_NUM\_CH $n$  + 1) times. (R/W)

**LEDC\_OVF\_CNT\_EN\_CH $n$**  This bit is used to enable the ovf\_cnt of channel  $n$ . (R/W)

**LEDC\_OVF\_CNT\_RESET\_CH $n$**  Set this bit to reset the ovf\_cnt of channel  $n$ . (WO)

**LEDC\_OVF\_CNT\_RESET\_ST\_CH $n$**  This is the status bit of LEDC\_OVF\_CNT\_RESET\_CH $n$ . (RO)

**Register 9.2: LEDC\_CH<sub>n</sub>\_CONF1\_REG (*n*: 0-7) (0x000C+0x14\**n*)**

LEDC_DUTY_START_CH <sub>n</sub> LEDC_DUTY_INC_CH <sub>n</sub>		LEDC_DUTY_NUM_CH <sub>n</sub>		LEDC_DUTY_CYCLE_CH <sub>n</sub>		LEDC_DUTY_SCALE_CH <sub>n</sub>	
31	30	29	20	19	10	9	0
0	1	0x0	0x0	0x0	0x0	0x0	Reset

**LEDC\_DUTY\_SCALE\_CH<sub>n</sub>** This register is used to configure the changing step scale of duty on channel *n*. (R/W)

**LEDC\_DUTY\_CYCLE\_CH<sub>n</sub>** The duty will change every LEDC\_DUTY\_CYCLE\_CH<sub>n</sub> on channel *n*. (R/W)

**LEDC\_DUTY\_NUM\_CH<sub>n</sub>** This register is used to control the number of times the duty cycle will be changed. (R/W)

**LEDC\_DUTY\_INC\_CH<sub>n</sub>** This register is used to increase or decrease the duty of output signal on channel *n*. 1: Increase;0: Decrease. (R/W)

**LEDC\_DUTY\_START\_CH<sub>n</sub>** Other configured fields in LEDC\_CH<sub>n</sub>\_CONF1\_REG will start to take effect when this bit is set to 1. (R/W)

**Register 9.3: LEDC\_CONF\_REG (0x00D0)**

LEDC_CLK_EN		(reserved)		LEDC_APB_CLK_SEL	
31	30	2	1	0	
0	0	0	0	0	0x0

**LEDC\_APB\_CLK\_SEL** This bit is used to select clock source for the 4 timers .

2'd1: APB\_CLK 2'd2: RTC8M\_CLK 2'd3: XTAL\_CLK (R/W)

**LEDC\_CLK\_EN** This bit is used to control clock.

1'b1: Force clock on for register. 1'h0: Support clock only when application writes registers. (R/W)

**Register 9.4: LEDC\_CH<sub>*n*</sub>\_HPOINT\_REG (*n*: 0-7) (0x0004+0x14\**n*)**

(reserved)														LEDC_HPOINT_CH <sup>n</sup>																		
31														14	13																	0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0														0x00																	Reset	

**LEDC\_HPOINT\_CH<sub>*n*</sub>** The output value changes to high when the selected timers has reached the value specified by this register. (R/W)

**Register 9.5: LEDC\_CH<sub>*n*</sub>\_DUTY\_REG (*n*: 0-7) (0x0008+0x14\**n*)**

(reserved)														LEDC_DUTY_CH <sup>n</sup>																																										
31														19														18																	0											
0														0														0														0x000														Reset

**LEDC\_DUTY\_CH<sub>*n*</sub>** This register is used to change the output duty by controlling the Lpoint.  
The output value turns to low when the selected timers has reached the Lpoint. (R/W)

**Register 9.6: LEDC\_CH<sub>*n*</sub>\_DUTY\_R\_REG (*n*: 0-7) (0x0010+0x14\**n*)**

(reserved)														LEDC_DUTY_R_CH <sup>n</sup>																															
31														19														18																	0
0 0 0 0 0 0 0 0 0 0 0 0 0 0														0x000																	Reset														

**LEDC\_DUTY\_R\_CH<sub>*n*</sub>** This register stores the current duty of output signal on channel *n*. (RO)

**Register 9.7: LEDC\_TIMER<sub>x</sub>\_CONF\_REG (x: 0-3) (0x00A0+0x8\*x)**

(reserved)										LEDC_TIMER <del>x</del> _PARAM_UP LEDC_TIMER <del>x</del> _TICK_SEL_TIMER <del>x</del> LEDC_TIMER <del>x</del> _RST LEDC_TIMER <del>x</del> _PAUSE										LEDC_CLK_DIV_TIMER <del>x</del>										LEDC_TIMER <del>x</del> _DUTY_RES																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
31						26						25	24	23	22	21						4						3	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0						0					

**LEDC\_TIMER<sub>x</sub>\_DUTY\_RES** This register is used to control the range of the counter in timer <sub>x</sub>. (R/W)

**LEDC\_CLK\_DIV\_TIMER<sub>x</sub>** This register is used to configure the divisor for the divider in timer <sub>x</sub>.

The least significant eight bits represent the fractional part. (R/W)

**LEDC\_TIMER<sub>x</sub>\_PAUSE** This bit is used to suspend the counter in timer <sub>x</sub>. (R/W)

**LEDC\_TIMER<sub>x</sub>\_RST** This bit is used to reset timer <sub>x</sub>. The counter will show 0 after reset. (R/W)

**LEDC\_TICK\_SEL\_TIMER<sub>x</sub>** This bit is used to select clock for timer <sub>x</sub>. When this bit is set to 1 LEDC\_APB\_CLK\_SEL[1:0] should be 1, otherwise the timer clock may be not accurate.

1'h0: SLOW\_CLK 1'h1: REF\_TICK (R/W)

**LEDC\_TIMER<sub>x</sub>\_PARAM\_UP** Set this bit to update LEDC\_CLK\_DIV\_TIMER<sub>x</sub> and LEDC\_TIMER<sub>x</sub>\_DUTY\_RES. (WO)

**Register 9.8: LEDC\_TIMER<sub>x</sub>\_VALUE\_REG (x: 0-3) (0x00A4+0x8\*x)**

(reserved)														LEDC_TIMER <sub>x</sub> _CNT																																									
31														14														13														0													
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0														0x00														Reset																											

**LEDC\_TIMER<sub>x</sub>\_CNT** This register stores the current counter value of timer <sub>x</sub>. (RO)

Register 9.9: LEDC\_INT\_RAW\_REG (0x00C0)

(reserved)																				LEDC_OVF_CNT_CH7_INT_RAW LEDC_OVF_CNT_CH6_INT_RAW LEDC_OVF_CNT_CH5_INT_RAW LEDC_OVF_CNT_CH4_INT_RAW LEDC_OVF_CNT_CH3_INT_RAW LEDC_OVF_CNT_CH2_INT_RAW LEDC_OVF_CNT_CH1_INT_RAW LEDC_OVF_CNT_CH0_INT_RAW LEDC_DUTY_CHNG_END_CH7_INT_RAW LEDC_DUTY_CHNG_END_CH6_INT_RAW LEDC_DUTY_CHNG_END_CH5_INT_RAW LEDC_DUTY_CHNG_END_CH4_INT_RAW LEDC_DUTY_CHNG_END_CH3_INT_RAW LEDC_DUTY_CHNG_END_CH2_INT_RAW LEDC_DUTY_CHNG_END_CH1_INT_RAW LEDC_DUTY_CHNG_END_CH0_INT_RAW LEDC_TIMER3_OVF_INT_RAW LEDC_TIMER2_OVF_INT_RAW LEDC_TIMER1_OVF_INT_RAW LEDC_TIMER0_OVF_INT_RAW																				
31																				20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset										

**LEDC\_TIMER<sub>x</sub>\_OVF\_INT\_RAW** Triggered when the timer<sub>x</sub> has reached its maximum counter value. (RO)

**LEDC\_DUTY\_CHNG\_END\_CH<sub>n</sub>\_INT\_RAW** Interrupt raw bit for channel <sub>n</sub>. Triggered when the gradual change of duty has finished. (RO)

**LEDC\_OVF\_CNT\_CH<sub>n</sub>\_INT\_RAW** Interrupt raw bit for channel <sub>n</sub>. Triggered when the ovf\_cnt has reached the value specified by LEDC\_OVF\_NUM\_CH<sub>n</sub>. (RO)

Register 9.10: LEDC\_INT\_ST\_REG (0x00C4)

(reserved)																						LEDC_OVF_CNT_CH7_INT_ST												LEDC_OVF_CNT_CH6_INT_ST												LEDC_OVF_CNT_CH5_INT_ST												LEDC_OVF_CNT_CH4_INT_ST												LEDC_OVF_CNT_CH3_INT_ST												LEDC_OVF_CNT_CH2_INT_ST												LEDC_OVF_CNT_CH1_INT_ST												LEDC_OVF_CNT_CH0_INT_ST												LEDC_DUTY_CHNG_END_CH7_INT_ST												LEDC_DUTY_CHNG_END_CH6_INT_ST												LEDC_DUTY_CHNG_END_CH5_INT_ST												LEDC_DUTY_CHNG_END_CH4_INT_ST												LEDC_DUTY_CHNG_END_CH3_INT_ST												LEDC_DUTY_CHNG_END_CH2_INT_ST												LEDC_DUTY_CHNG_END_CH1_INT_ST												LEDC_DUTY_CHNG_END_CH0_INT_ST												LEDC_TIMER3_OVF_INT_ST												LEDC_TIMER2_OVF_INT_ST												LEDC_TIMER1_OVF_INT_ST												LEDC_TIMER0_OVF_INT_ST																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
31												20												19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**LEDC\_TIMER<sub>x</sub>\_OVF\_INT\_ST** This is the masked interrupt status bit for the LEDC\_TIMER<sub>x</sub>\_OVF\_INT interrupt when LEDC\_TIMER<sub>x</sub>\_OVF\_INT\_ENA is set to 1. (RO)

**LEDC\_DUTY\_CHNG\_END\_CH<sub>n</sub>\_INT\_ST** This is the masked interrupt status bit for the LEDC\_DUTY\_CHNG\_END\_CH<sub>n</sub>\_INT interrupt when LEDC\_DUTY\_CHNG\_END\_CH<sub>n</sub>\_INT\_ENAIS set to 1. (RO)

**LEDC\_OVF\_CNT\_CH<sub>n</sub>\_INT\_ST** This is the masked interrupt status bit for the LEDC\_OVF\_CNT\_CH<sub>n</sub>\_INT interrupt when LEDC\_OVF\_CNT\_CH<sub>n</sub>\_INT\_ENA is set to 1. (RO)

Register 9.11: LEDC\_INT\_ENA\_REG (0x00C8)

(reserved)																				LEDC_OVF_CNT_CH7_INT_ENA LEDC_OVF_CNT_CH6_INT_ENA LEDC_OVF_CNT_CH5_INT_ENA LEDC_OVF_CNT_CH4_INT_ENA LEDC_OVF_CNT_CH3_INT_ENA LEDC_OVF_CNT_CH2_INT_ENA LEDC_OVF_CNT_CH1_INT_ENA LEDC_DUTY_CHNG_END_CH0_INT_ENA LEDC_DUTY_CHNG_END_CH7_INT_ENA LEDC_DUTY_CHNG_END_CH6_INT_ENA LEDC_DUTY_CHNG_END_CH5_INT_ENA LEDC_DUTY_CHNG_END_CH4_INT_ENA LEDC_DUTY_CHNG_END_CH3_INT_ENA LEDC_DUTY_CHNG_END_CH2_INT_ENA LEDC_DUTY_CHNG_END_CH1_INT_ENA LEDC_TIMER3_OVF_INT_ENA LEDC_TIMER2_OVF_INT_ENA LEDC_TIMER1_OVF_INT_ENA LEDC_TIMER0_OVF_INT_ENA																					
31																				20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0												0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**LEDC\_TIMER<sub>x</sub>\_OVF\_INT\_ENA** The interrupt enable bit for the LEDC\_TIMER<sub>x</sub>\_OVF\_INT interrupt. (R/W)

**LEDC\_DUTY\_CHNG\_END\_CH<sub>n</sub>\_INT\_ENA** The interrupt enable bit for the LEDC\_DUTY\_CHNG\_END\_CH<sub>n</sub>\_INT interrupt. (R/W)

**LEDC\_OVF\_CNT\_CH<sub>n</sub>\_INT\_ENA** The interrupt enable bit for the LEDC\_OVF\_CNT\_CH<sub>n</sub>\_INT interrupt. (R/W)

Register 9.12: LEDC\_INT\_CLR\_REG (0x00CC)

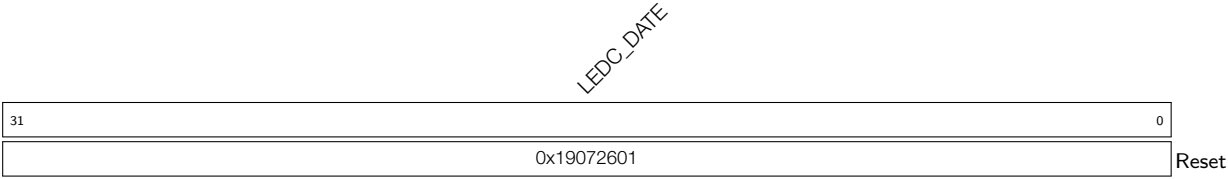
<div>(reserved)</div>																												<div>LEDC_OVF_CNT_CH7_INT_CLR LEDC_OVF_CNT_CH6_INT_CLR LEDC_OVF_CNT_CH5_INT_CLR LEDC_OVF_CNT_CH4_INT_CLR LEDC_OVF_CNT_CH3_INT_CLR LEDC_OVF_CNT_CH2_INT_CLR LEDC_OVF_CNT_CH1_INT_CLR LEDC_DUTY_CHNG_END_CH0_INT_CLR LEDC_DUTY_CHNG_END_CH7_INT_CLR LEDC_DUTY_CHNG_END_CH6_INT_CLR LEDC_DUTY_CHNG_END_CH5_INT_CLR LEDC_DUTY_CHNG_END_CH4_INT_CLR LEDC_DUTY_CHNG_END_CH3_INT_CLR LEDC_TIMER3_OVF_INT_CLR LEDC_TIMER2_OVF_INT_CLR LEDC_TIMER1_OVF_INT_CLR LEDC_TIMER0_OVF_INT_CLR</div>																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
31																				20				19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0			

**LEDC\_TIMER<sub>x</sub>\_OVF\_INT\_CLR** Set this bit to clear the LEDC\_TIMER<sub>x</sub>\_OVF\_INT interrupt. (WO)

**LEDC\_DUTY\_CHNG\_END\_CH<sub>n</sub>\_INT\_CLR** Set this bit to clear the LEDC\_DUTY\_CHNG\_END\_CH<sub>n</sub>\_INT interrupt. (WO)

**LEDC\_OVF\_CNT\_CH<sub>n</sub>\_INT\_CLR** Set this bit to clear the LEDC\_OVF\_CNT\_CH<sub>n</sub>\_INT interrupt. (WO)

Register 9.13: LEDC\_DATE\_REG (0x00FC)



**LEDC\_DATE** This is the version control register. (R/W)

## 10. Remote Control Peripheral

### 10.1 Introduction

The RMT (Remote Control) module is designed to send/receive infrared remote control signals that support for a variety of remote control protocols. The RMT module converts pulse codes stored in the module's built-in RAM into output signals, or converts input signals into pulse codes and stores them back in RAM. In addition, the RMT module optionally modulates its output signals with a carrier wave, or optionally filters its input signals.

The RMT module has four channels, numbered from zero to three. Each channel has the same functionality controlled by dedicated set of registers and is able to independently transmit or receive data. Registers in each channel are indicated by  $n$  which is used as a placeholder for the channel number.

### 10.2 Functional Description

#### 10.2.1 RMT Architecture

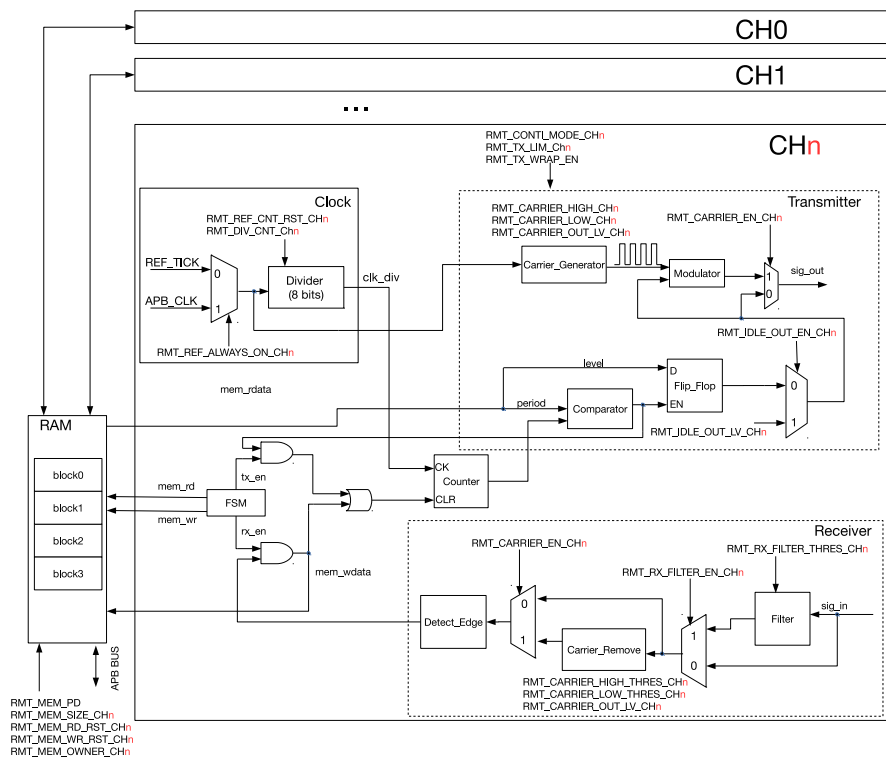
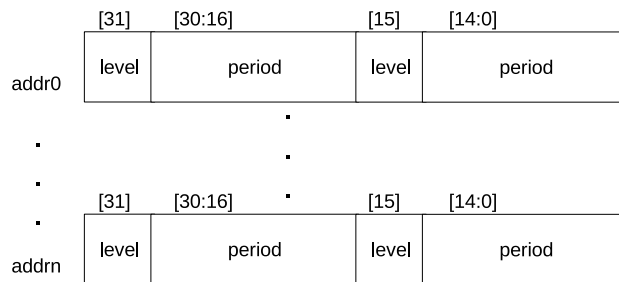


Figure 10-1. RMT Architecture

The RMT module contains four independent channels. Each channel has a clock divider, a counter, a transmitter and a receiver. As for the transmitter and the receiver of a single channel, only one of them can be active. The four channels share a 256 x 32-bit RAM.



## 10.2.2 RMT RAM



**Figure 10-2. Format of Pulse Code in RAM**

The format of pulse code in RAM is shown in Figure 10-2. Each pulse code contains a 16-bit entry with two fields, level and period. Level (0 or 1) indicates a high-/low-level value was received or is going to be sent, while "period" points out the clock cycles (see Figure 10-1 clk\_div) for which the level lasts. A zero period is interpreted as a transmission end-marker.

The RAM is divided into four 64 x 32-bit blocks. By default, each channel uses one block (block zero for channel zero, block one for channel one, and so on). Usually, only one block of 64 x 32-bit worth of data can be sent or received in channel  $n$ . If the data size is larger than this block size, users can configure the channel to enable wrap mode or to use more blocks by setting `RMT_MEM_SIZE_CH $n$` . Setting `RMT_MEM_SIZE_CH $n$`  > 1 will prompt channel  $n$  to use the memory of subsequent channels, block ( $n$ ) ~ block ( $n + \text{RMT\_MEM\_SIZE\_CH}_n - 1$ ). If so, the subsequent channels  $n + 1 \sim n + \text{RMT\_MEM\_SIZE\_CH}_n - 1$  cannot be used once their RAM blocks are occupied. Note that the RAM used by each channel is mapped from low address to high address. In such mode, channel 0 is able to use the RAM blocks for channels 1, 2 and 3 by setting `RMT_MEM_SIZE_CH $n$` , but channel 3 cannot use the blocks for channels 0, 1, or 2.

The RMT RAM can be accessed via APB bus, or read by the transmitter and written by the receiver. To protect a receiver from overwriting the blocks a transmitter is about to transmit, `RMT_MEM_OWNER_CH $n$`  can be configured to designate the block's owner, be it a transmitter or receiver. This way, if this ownership is violated, an `RMT_MEM_OWNER_ERR_CH $n$`  flag will be generated.

When the RMT module is inactive, the RAM can be put into low-power mode by setting `RMT_MEM_FORCE_PD`.

## 10.2.3 Clock

The drive clock of a divider is generated by taking either the APB\_CLK or REF\_TICK according to the state of `RMT_REF_ALWAYS_ON_CH $n$` . (For more information on clock sources, please see Chapter [Reset and Clock](#)). Divider value is normally equal to the value of `RMT_DIV_CNT_CH $n$` , except value 0 that represents divider 256. The clock divider can be reset to zero by clearing `RMT_REF_CNT_RST_CH $n$` . The clock generated from the divider can be used by the clock counter (see Figure 10-2).

## 10.2.4 Transmitter

When `RMT_TX_START_CH $n$`  is set to 1, the transmitter of channel  $n$  will start reading and sending pulse codes from the starting address of its RAM block. The codes are sent starting from low-address entry. The transmitter will stop the transmission, return to idle state and generate an `RMT_CH $n$ _TX_END_INT` interrupt, when an end-marker (a zero period) is encountered. Also, setting `RMT_TX_STOP_CH $n$`  to 1 stops the transmission and immediately sets the transmitter back to idle. The output level of a transmitter in idle state is determined by the "level" field of the end-marker or by the content of `RMT_IDLE_OUT_LV_CH $n$` , depending on the configuration of

### RMT\_IDLE\_OUT\_EN\_CH $n$ .

To transmit more pulse codes than can be fitted in the channel's RAM, users can enable wrap mode by configuring `RMT_MEM_TX_WRAP_EN`. In this mode, when the transmitter has reached the end-marker in the channel's memory, it will loop back to the first byte. For example, if `RMT_MEM_SIZE_CH $n$`  is set to 1, the transmitter will start sending data from the address  $64 * n$ , and then the data from higher RAM address. Once the transmitter finishes sending the data from  $(64 * (n + 1) - 1)$ , it will continue sending data from  $64 * n$  till encounters an end-marker. Wrap mode is also applicable for `RMT_MEM_SIZE_CH $n$`  > 1.

An `RMT_CH $n$ _TX_THR_EVENT_INT` interrupt will be generated whenever the size of transmitted pulse codes is larger than or equal to the value set by `RMT_TX_LIM_CH $n$` . In wrap mode, `RMT_TX_LIM_CH $n$`  can be set to a half or a fraction of the size of the channel's RAM block. When an `RMT_CH $n$ _TX_THR_EVENT_INT` interrupt is detected, the already used RAM region should be updated by subsequent user defined events. Therefore, when the wrap mode happens the transmitter will seamlessly continue sending the new events.

The output of the transmitter can be modulated using a carrier wave by setting `RMT_CARRIER_EN_CH $n$` . The carrier waveform is configurable. In a carrier cycle, the high level lasts for  $(\text{RMT\_CARRIER\_HIGH\_CH}_n + 1)$  clock cycles of APB\_CLK or REF\_TICK, while the low level lasts for  $(\text{RMT\_CARRIER\_LOW\_CH}_n + 1)$  clock cycles of APB\_CLK or REF\_TICK. When `RMT_CARRIER_OUT_LV_CH $n$`  is set to 1, carrier wave will be added on high-level output signals; while `RMT_CARRIER_OUT_LV_CH $n$`  is set to 0, carrier wave will be added on low-level output signals. Carrier wave can be added on output signals during modulation, or just added on valid pulse codes (the data stored in RAM), which can be set by configuring `RMT_CARRIER_EFF_EN_CH $n$` .

The continuous transmission of the transmitter can be enabled by setting `RMT_TX_CONTI_MODE_CH $n$` . When this register is set, the transmitter will send the pulse codes from RAM in loops. If `RMT_TX_LOOP_CNT_EN_CH $n$`  is set to 1, the transmitter will start counting loop times. Once the counting reaches the value of register `RMT_TX_LOOP_NUM_CH $n$` , an `RMT_CH $n$ _TX_LOOP_INT` interrupt will be generated.

Setting `RMT_TX_SIM_EN` to 1 will enable multiple channels to start sending data simultaneously. `RMT_TX_SIM_CH $n$`  will choose which multiple channels are enabled to send data simultaneously.

## 10.2.5 Receiver

When `RMT_RX_EN_CH $n$`  is set to 1, the receiver in channel  $n$  becomes active, detecting signal levels and measuring clock cycles the signals lasts for. These data will be written in RAM in the form of pulse codes. Receiving ends, when the receiver detects no change in a signal level for a number of clock cycles more than the value set by `RMT_IDLE_THRES_CH $n$` . The receiver will return to idle state and generate an `RMT_CH $n$ _RX_END_INT` interrupt.

The receiver has an input signal filter which can be enabled by configuring `RMT_RX_FILTER_EN_CH $n$` . The filter samples input signals continuously, and will detect the signals which remain unchanged for a continuous `RMT_RX_FILTER_THRES_CH $n$`  APB clock cycles as valid, otherwise, the signals will be detected as invalid. Only the valid signals can pass through this filter. The filter will remove pulses with a length of less than `RMT_RX_FILTER_THRES_CH $n$`  APB clock cycles.

## 10.2.6 Interrupts

- `RMT_CH $n$ _ERR_INT`: Triggered when channel  $n$  does not read or write data correctly. For example, if the transmitter still tries to read data from RAM when the RAM is empty, or the receiver still tries to write data into RAM when the RAM is full, this interrupt will be triggered.
- `RMT_CH $n$ _TX_THR_EVENT_INT`: Triggered when the amount of data the transmitter has sent matches the

value of `RMT_TX_LIM_CH $n$` .

- `RMT_CH $n$ _TX_END_INT`: Triggered when the transmitter has finished transmitting signals.
- `RMT_CH $n$ _RX_END_INT`: Triggered when the receiver has finished receiving signals.
- `RMT_CH $n$ _TX_LOOP_INT`: Triggered when the loop counting reaches the value set by `RMT_TX_LOOP_NUM_CH $n$` .

### 10.3 Base Address

Users can access RMT with two base addresses, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter 1: *System and Memory*.

**Table 45: RMT Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F416000
PeriBUS2	0x60016000

### 10.4 Register Summary

The addresses in the following table are relative to RMT base addresses provided in Section 10.3.

Name	Description	Address	Access
<b>Configuration registers</b>			
<code>RMT_CH0CONF0_REG</code>	Channel 0 configuration register 0	0x0010	R/W
<code>RMT_CH0CONF1_REG</code>	Channel 0 configuration register 1	0x0014	varies
<code>RMT_CH1CONF0_REG</code>	Channel 1 configuration register 0	0x0018	R/W
<code>RMT_CH1CONF1_REG</code>	Channel 1 configuration register 1	0x001C	varies
<code>RMT_CH2CONF0_REG</code>	Channel 2 configuration register 0	0x0020	R/W
<code>RMT_CH2CONF1_REG</code>	Channel 2 configuration register 1	0x0024	varies
<code>RMT_CH3CONF0_REG</code>	Channel 3 configuration register 0	0x0028	R/W
<code>RMT_CH3CONF1_REG</code>	Channel 3 configuration register 1	0x002C	varies
<code>RMT_APB_CONF_REG</code>	RMT APB configuration register	0x0080	R/W
<code>RMT_REF_CNT_RST_REG</code>	RMT clock divider reset register	0x0088	R/W
<code>RMT_CH0_RX_CARRIER_RM_REG</code>	Channel 0 carrier remove register	0x008C	R/W
<code>RMT_CH1_RX_CARRIER_RM_REG</code>	Channel 1 carrier remove register	0x0090	R/W
<code>RMT_CH2_RX_CARRIER_RM_REG</code>	Channel 2 carrier remove register	0x0094	R/W
<code>RMT_CH3_RX_CARRIER_RM_REG</code>	Channel 3 carrier remove register	0x0098	R/W
<b>Carrier wave duty cycle registers</b>			
<code>RMT_CH0CARRIER_DUTY_REG</code>	Channel 0 duty cycle configuration register	0x0060	R/W
<code>RMT_CH1CARRIER_DUTY_REG</code>	Channel 1 duty cycle configuration register	0x0064	R/W
<code>RMT_CH2CARRIER_DUTY_REG</code>	Channel 2 duty cycle configuration register	0x0068	R/W
<code>RMT_CH3CARRIER_DUTY_REG</code>	Channel 3 duty cycle configuration register	0x006C	R/W
<b>Tx event configuration registers</b>			
<code>RMT_CH0_TX_LIM_REG</code>	Channel 0 Tx event configuration register	0x0070	varies
<code>RMT_CH1_TX_LIM_REG</code>	Channel 1 Tx event configuration register	0x0074	varies
<code>RMT_CH2_TX_LIM_REG</code>	Channel 2 Tx event configuration register	0x0078	varies

Name	Description	Address	Access
<a href="#">RMT_CH3_TX_LIM_REG</a>	Channel 3 Tx event configuration register	0x007C	varies
<a href="#">RMT_TX_SIM_REG</a>	Enable RMT simultaneous transmission	0x0084	R/W
<b>Status registers</b>			
<a href="#">RMT_CH0STATUS_REG</a>	Channel 0 status register	0x0030	RO
<a href="#">RMT_CH1STATUS_REG</a>	Channel 1 status register	0x0034	RO
<a href="#">RMT_CH2STATUS_REG</a>	Channel 2 status register	0x0038	RO
<a href="#">RMT_CH3STATUS_REG</a>	Channel 3 status register	0x003C	RO
<a href="#">RMT_CH0ADDR_REG</a>	Channel 0 address register	0x0040	RO
<a href="#">RMT_CH1ADDR_REG</a>	Channel 1 address register	0x0044	RO
<a href="#">RMT_CH2ADDR_REG</a>	Channel 2 address register	0x0048	RO
<a href="#">RMT_CH3ADDR_REG</a>	Channel 3 address register	0x004C	RO
<b>Version register</b>			
<a href="#">RMT_DATE_REG</a>	Version control register	0x00FC	R/W
<b>FIFO R/W registers</b>			
<a href="#">RMT_CH0DATA_REG</a>	Read and write data for channel 0 via APB FIFO	0x0000	RO
<a href="#">RMT_CH1DATA_REG</a>	Read and write data for channel 1 via APB FIFO	0x0004	RO
<a href="#">RMT_CH2DATA_REG</a>	Read and write data for channel 2 via APB FIFO	0x0008	RO
<a href="#">RMT_CH3DATA_REG</a>	Read and write data for channel 3 via APB FIFO	0x000C	RO
<b>Interrupt registers</b>			
<a href="#">RMT_INT_RAW_REG</a>	Raw interrupt status register	0x0050	RO
<a href="#">RMT_INT_ST_REG</a>	Masked interrupt status register	0x0054	RO
<a href="#">RMT_INT_ENA_REG</a>	Interrupt enable register	0x0058	R/W
<a href="#">RMT_INT_CLR_REG</a>	Interrupt clear register	0x005C	WO

## 10.5 Registers

**Register 10.1: RMT\_CH $n$ CONF0\_REG ( $n$ : 0-3) (0x0010+8\* $n$ )**

(reserved)																RMT_CARRIER_OUT_LV_CH <sup>n</sup>																RMT_CARRIER_EN_CH <sup>n</sup>																RMT_CARRIER_EFF_EN_CH <sup>n</sup>																RMT_MEM_SIZE_CH <sup>n</sup>																RMT_IDLE_THRES_CH <sup>n</sup>																																																RMT_DIV_CNT_CH <sup>n</sup>																															
31	30	29	28	27	26											24	23											8	7											0																																																																																																																							
0	0	1	1	1	0x1										0x1000										0x2										Reset																																																																																																																												

**RMT\_DIV\_CNT\_CH $n$**  This field is used to configure clock divider for channel  $n$ . (R/W)

**RMT\_IDLE\_THRES\_CH $n$**  Receiving ends when no edge is detected on input signals for continuous clock cycles larger than this register value. (R/W)

**RMT\_MEM\_SIZE\_CH $n$**  This field is used to configure the maximum blocks allocated to channel  $n$ . The valid range is from 1 ~ 4- $n$ . (R/W)

**RMT\_CARRIER\_EFF\_EN\_CH $n$**  1: Add carrier modulation on output signals only at data sending state for channel  $n$ . 0: Add carrier modulation on signals at all states for channel  $n$ . States here include idle state(ST\_IDLE), reading data from RAM (ST\_RD\_MEM), and sending data stored in RAM (ST\_SEND). Only valid when **RMT\_CARRIER\_EN\_CH $n$**  is set to 1. (R/W)

**RMT\_CARRIER\_EN\_CH $n$**  This bit is used to enable carrier modulation for channel  $n$ . 1: Add carrier modulation on output signals. 0: No carrier modulation is added on output signals. (R/W)

**RMT\_CARRIER\_OUT\_LV\_CH $n$**  This bit is used to configure the position of carrier wave for channel  $n$ . 1'h0: Add carrier wave on low-level output signals. 1'h1: Add carrier wave on high-level output signals. (R/W)

**Register 10.2: RMT\_CH $n$ CONF1\_REG ( $n$ : 0-3) (0x0014+8\* $n$ )**

(reserved)																RMT_TX_STOP_CH <sup>n</sup>					RMT_IDLE_OUT_EN_CH <sup>n</sup>					RMT_IDLE_OUT_LV_CH <sup>n</sup>					RMT_REF_ALWAYS_ON_CH <sup>n</sup>					RMT_CHK_RX_CARRIER_EN_CH <sup>n</sup>					RMT_RX_FILTER_THRES_CH <sup>n</sup>					RMT_RX_FILTER_EN_CH <sup>n</sup>					RMT_TX_CONTL_MODE_CH <sup>n</sup>					RMT_MEM_OWNER_CH <sup>n</sup>					RMT_APB_MEM_RST_CH <sup>n</sup>					RMT_MEM_RD_RST_CH <sup>n</sup>					RMT_MEM_WR_RST_CH <sup>n</sup>					RMT_RX_EN_CH <sup>n</sup>					RMT_TX_START_CH <sup>n</sup>				
31																21					20	19	18	17	16	15	8																7	6	5	4	3	2	1	0	Reset																																		
0																0	0	0	0	0	0	0	0	0	0xf																0	0	1	0	0	0	0	0	0																																				

**RMT\_TX\_START\_CH $n$**  Set this bit to start sending data on channel  $n$ . (R/W)

**RMT\_RX\_EN\_CH $n$**  Set this bit to enable receiver to receive data on channel  $n$ . (R/W)

**RMT\_MEM\_WR\_RST\_CH $n$**  Set this bit to reset RAM write address accessed by the receiver for channel  $n$ . (WO)

**RMT\_MEM\_RD\_RST\_CH $n$**  Set this bit to reset RAM read address accessed by the transmitter for channel  $n$ . (WO)

**RMT\_MEM\_OWNER\_CH $n$**  This bit marks the ownership of channel  $n$ 's RAM block. 1'h1: Receiver is using the RAM. 1'h0: Transmitter is using the RAM. (R/W)

**RMT\_TX\_CONTI\_MODE\_CH $n$**  Set this bit to restart transmission in continuous mode from the first data in channel  $n$ . (R/W)

**RMT\_RX\_FILTER\_EN\_CH $n$**  Set this bit to enable the receiver's filter for channel  $n$ . (R/W)

**RMT\_RX\_FILTER\_THRES\_CH $n$**  Set this field to ignore the input pulse when its width is less than RMT\_RX\_FILTER\_THRES\_CH $n$  APB clock cycles in receive mode. (R/W)

**RMT\_CHK\_RX\_CARRIER\_EN\_CH $n$**  Set this bit to enable memory loop read mode when carrier modulation is enabled for channel  $n$ . (R/W)

**RMT\_REF\_ALWAYS\_ON\_CH $n$**  Set this bit to select a base clock for channel  $n$ . 1'h1: APB\_CLK; 1'h0: REF\_TICK (R/W)

**RMT\_IDLE\_OUT\_LV\_CH $n$**  This bit configures the level of output signals in channel  $n$  when the transmitter is in idle state. (R/W)

**RMT\_IDLE\_OUT\_EN\_CH $n$**  This is the output enable bit for channel  $n$  in idle state. (R/W)

**RMT\_TX\_STOP\_CH $n$**  Set this bit to stop the transmitter of channel  $n$  sending data out. (R/W)

Register 10.3: RMT\_APB\_CONF\_REG (0x0080)

RMT_CLK_EN		(reserved)																				RMT_MEM_FORCE_PU					RMT_MEM_FORCE_PD		RMT_MEM_CLK_FORCE_ON		RMT_MEM_TX_WRAP_EN		RMT_APB_FIFO_MASK	
31	30																					5	4	3	2	1	0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	Reset				

Reset

**RMT\_APB\_FIFO\_MASK** 1'h1: Access memory directly; 1'h0: Access memory via APB FIFO. (R/W)

**RMT\_MEM\_TX\_WRAP\_EN** Set this bit to enable wrap mode. (R/W)

**RMT\_MEM\_CLK\_FORCE\_ON** Set this bit to enable the clock for RAM when RMT module starts working; disable this clock when RMT stops working, to achieve low-power scheme. (R/W)

**RMT\_MEM\_FORCE\_PD** Set this bit to power down RMT memory. (R/W)

**RMT\_MEM\_FORCE\_PU** 1: Disable RAM's Light-sleep power down function. 0: power down RMT RAM when RMT is in Light-sleep mode. (R/W)

**RMT\_CLK\_EN** Clock gating enable bit for RMT registers to achieve low-power scheme. 1: Power up drive clock for RMT registers. 0: Power down drive clock for RMT registers. (R/W)

Register 10.4: RMT\_REF\_CNT\_RST\_REG (0x0088)

(reserved)																												RMT_REF_CNT_RST_CH3					RMT_REF_CNT_RST_CH2					RMT_REF_CNT_RST_CH1					RMT_REF_CNT_RST_CH0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
31																												4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

**RMT\_REF\_CNT\_RST\_CH $n$**  This bit is used to reset the clock divider of channel  $n$ . (R/W)

**Register 10.5: RMT\_CH $n$ \_RX\_CARRIER\_RM\_REG ( $n$ : 0-3) (0x008C+4\* $n$ )**

RMT_CARRIER_HIGH_THRES_CH <sup>n</sup>																RMT_CARRIER_LOW_THRES_CH <sup>n</sup>																
31																16	15														0	
0x00																0x00																Reset

**RMT\_CARRIER\_LOW\_THRES\_CH $n$**  The low level period in carrier modulation mode is (RMT\_CARRIER\_LOW\_THRES\_CH $n$  + 1) clock cycles for channel  $n$ . (R/W)

**RMT\_CARRIER\_HIGH\_THRES\_CH $n$**  The high level period in carrier modulation mode is (RMT\_CARRIER\_HIGH\_THRES\_CH $n$  + 1) clock cycles for channel  $n$ . (R/W)

**Register 10.6: RMT\_CH $n$ CARRIER\_DUTY\_REG ( $n$ : 0-3) (0x0060+4\* $n$ )**

RMT_CARRIER_HIGH_CH <sup>n</sup>																RMT_CARRIER_LOW_CH <sup>n</sup>																
31																16	15														0	
0x40																0x40																Reset

**RMT\_CARRIER\_LOW\_CH $n$**  This field is used to configure the clock cycles of carrier wave at low level for channel  $n$ . (R/W)

**RMT\_CARRIER\_HIGH\_CH $n$**  This field is used to configure the clock cycles of carrier wave at high level for channel  $n$ . (R/W)





**Register 10.9: RMT\_CH $n$ STATUS\_REG ( $n$ : 0-3) (0x0030+4\* $n$ )**

(reserved)				RMT_APB_MEM_PD_ERR_CH <sub>n</sub>				RMT_APB_MEM_IWR_ERR_CH <sub>n</sub>				RMT_MEM_EMPTY_CH <sub>n</sub>				RMT_MEM_FULL_CH <sub>n</sub>				RMT_MEM_OWNER_ERR_CH <sub>n</sub>				RMT_STATE_CH <sub>n</sub>				(reserved)				RMT_MEM_RADDR_EX_CH <sub>n</sub>				(reserved)				RMT_MEM_WADDR_EX_CH <sub>n</sub>			
31	28	27	26	25	24	23	22	20	19	18					10	9	8					0																					
0	0	0	0	0	0	0	0	0	0	0	0x0	0	0x0				0	0x0				Reset																					

**RMT\_MEM\_WADDR\_EX\_CH $n$**  This field records the memory address offset when receiver of channel  $n$  is using the RAM. (RO)

**RMT\_MEM\_RADDR\_EX\_CH $n$**  This field records the memory address offset when transmitter of channel  $n$  is using the RAM. (RO)

**RMT\_STATE\_CH $n$**  This field records the FSM status of channel  $n$ . (RO)

**RMT\_MEM\_OWNER\_ERR\_CH $n$**  This status bit will be set when the ownership of memory block is violated. (RO)

**RMT\_MEM\_FULL\_CH $n$**  This status bit will be set if the receiver receives more data than the memory allows. (RO)

**RMT\_MEM\_EMPTY\_CH $n$**  This status bit will be set when the data to be sent is more than the memory allows and the wrap mode is disabled. (RO)

**RMT\_APB\_MEM\_WR\_ERR\_CH $n$**  This status bit will be set if the offset address is out of memory size when channel  $n$  writes RAM via APB bus. (RO)

**RMT\_APB\_MEM\_RD\_ERR\_CH $n$**  This status bit will be set if the offset address is out of memory size when channel  $n$  reads RAM via APB bus. (RO)

**Register 10.10: RMT\_CH $n$ ADDR\_REG ( $n$ : 0-3) (0x0040+4\* $n$ )**

(reserved)																RMT_APB_MEM_RADDR_CH <sup>n</sup>																(reserved)																RMT_APB_MEM_WADDR_CH <sup>n</sup>																																																																																																															
31																19																18																10																9																8																0																																																															
0																0																0																0																0																0																0x0																0																0x0																Reset															

**RMT\_APB\_MEM\_WADDR\_CH $n$**  This field records the memory address offset when channel  $n$  writes RAM via APB bus. (RO)

**RMT\_APB\_MEM\_RADDR\_CH $n$**  This field records the memory address offset when channel  $n$  reads RAM via APB bus. (RO)

## Register 10.11: RMT\_DATE\_REG (0x00FC)

RMT_DATE	
31	0
0x19072601	
Reset	

**RMT\_DATE** Version control register. (R/W)

Register 10.12: RMT\_CH $n$ DATA\_REG ( $n$ : 0-3) (0x0000+4\* $n$ )

RMT_CH <sub>n</sub> _DATA_REG	
31	0
0x000000	
Reset	

**RMT\_CH $n$ DATA\_REG** This register is used to read and write data for channel  $n$  via APB FIFO. (RO)

## Register 10.13: RMT\_INT\_RAW\_REG (0x0050)

(reserved)															
RMT_CH3_TX_LOOP_INT_RAW RMT_CH2_TX_LOOP_INT_RAW RMT_CH1_TX_LOOP_INT_RAW RMT_CH0_TX_LOOP_INT_RAW RMT_CH3_TX_THR_EVENT_INT_RAW RMT_CH2_TX_THR_EVENT_INT_RAW RMT_CH1_TX_THR_EVENT_INT_RAW RMT_CH0_TX_THR_EVENT_INT_RAW RMT_CH3_ERR_INT_RAW RMT_CH2_ERR_INT_RAW RMT_CH1_ERR_INT_RAW RMT_CH0_ERR_INT_RAW RMT_CH3_TX_END_INT_RAW RMT_CH2_TX_END_INT_RAW RMT_CH1_TX_END_INT_RAW RMT_CH0_TX_END_INT_RAW RMT_CH3_RX_END_INT_RAW RMT_CH2_RX_END_INT_RAW RMT_CH1_RX_END_INT_RAW RMT_CH0_RX_END_INT_RAW															
31															0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset															

**RMT\_CH $n$ TX\_END\_INT\_RAW** The interrupt raw bit for channel  $n$ . Triggered when transmitting ends. (RO)

**RMT\_CH $n$ RX\_END\_INT\_RAW** The interrupt raw bit for channel  $n$ . Triggered when receiving ends. (RO)

**RMT\_CH $n$ ERR\_INT\_RAW** The interrupt raw bit for channel  $n$ . Triggered when error occurs. (RO)

**RMT\_CH $n$ TX\_THR\_EVENT\_INT\_RAW** The interrupt raw bit for channel  $n$ . Triggered when transmitter sends more data than configured value. (RO)

**RMT\_CH $n$ TX\_LOOP\_INT\_RAW** The interrupt raw bit for channel  $n$ . Triggered when loop counting reaches the configured threshold value. (RO)

Register 10.14: RMT\_INT\_ST\_REG (0x0054)

(reserved)																RMT_CH3_TX_LOOP_INT_ST RMT_CH2_TX_LOOP_INT_ST RMT_CH1_TX_LOOP_INT_ST RMT_CH0_TX_LOOP_INT_ST RMT_CH3_TX_THR_EVENT_INT_ST RMT_CH2_TX_THR_EVENT_INT_ST RMT_CH1_TX_THR_EVENT_INT_ST RMT_CH0_TX_THR_EVENT_INT_ST RMT_CH3_RX_END_INT_ST RMT_CH2_RX_END_INT_ST RMT_CH1_RX_END_INT_ST RMT_CH0_RX_END_INT_ST RMT_CH3_ERR_INT_ST RMT_CH2_ERR_INT_ST RMT_CH1_ERR_INT_ST RMT_CH0_ERR_INT_ST RMT_CH0_TX_END_INT_ST																
31												20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset		

**RMT\_CH<sub>n</sub>\_TX\_END\_INT\_ST** The masked interrupt status bit for **RMT\_CH<sub>n</sub>\_TX\_END\_INT**. (RO)

**RMT\_CH<sub>n</sub>\_RX\_END\_INT\_ST** The masked interrupt status bit for **RMT\_CH<sub>n</sub>\_RX\_END\_INT**. (RO)

**RMT\_CH<sub>n</sub>\_ERR\_INT\_ST** The masked interrupt status bit for **RMT\_CH<sub>n</sub>\_ERR\_INT**. (RO)

**RMT\_CH<sub>n</sub>\_TX\_THR\_EVENT\_INT\_ST** The masked interrupt status bit for **RMT\_CH<sub>n</sub>\_TX\_THR\_EVENT\_INT**. (RO)

**RMT\_CH<sub>n</sub>\_TX\_LOOP\_INT\_ST** The masked interrupt status bit for **RMT\_CH<sub>n</sub>\_TX\_LOOP\_INT**. (RO)

Register 10.15: RMT\_INT\_ENA\_REG (0x0058)

(reserved)																			RMT_CH3_TX_LOOP_INT_ENA RMT_CH2_TX_LOOP_INT_ENA RMT_CH1_TX_LOOP_INT_ENA RMT_CH0_TX_LOOP_INT_ENA RMT_CH3_TX_THR_EVENT_INT_ENA RMT_CH2_TX_THR_EVENT_INT_ENA RMT_CH1_TX_THR_EVENT_INT_ENA RMT_CH0_TX_THR_EVENT_INT_ENA RMT_CH3_RX_END_INT_ENA RMT_CH2_RX_END_INT_ENA RMT_CH1_RX_END_INT_ENA RMT_CH0_RX_END_INT_ENA RMT_CH0_TX_END_INT_ENA																								
31																				20				19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset								

**RMT\_CH<sub>n</sub>\_TX\_END\_INT\_ENA** Interrupt enable bit for **RMT\_CH<sub>n</sub>\_TX\_END\_INT**. (R/W)

**RMT\_CH<sub>n</sub>\_RX\_END\_INT\_ENA** Interrupt enable bit for **RMT\_CH<sub>n</sub>\_RX\_END\_INT**. (R/W)

**RMT\_CH<sub>n</sub>\_ERR\_INT\_ENA** Interrupt enable bit for **RMT\_CH<sub>n</sub>\_ERR\_INT**. (R/W)

**RMT\_CH<sub>n</sub>\_TX\_THR\_EVENT\_INT\_ENA** Interrupt enable bit for **RMT\_CH<sub>n</sub>\_TX\_THR\_EVENT\_INT**. (R/W)

**RMT\_CH<sub>n</sub>\_TX\_LOOP\_INT\_ENA** Interrupt enable bit for **RMT\_CH<sub>n</sub>\_TX\_LOOP\_INT**. (R/W)

## 293

[Submit Documentation Feedback](#)

ESP32-S2 TRM (Preliminary V0.5)

## 11. Pulse Count Controller

The pulse count controller (PCNT) is designed to count input pulses and generate interrupts. It can increment or decrement a pulse counter value by keeping track of rising (positive) or falling (negative) edges of the input pulse signal. The PCNT has four independent pulse counters, called units which have their groups of registers. In this chapter,  $n$  denotes the number of a unit from 0 ~ 3.

Each unit includes two channels (ch0 and ch1) which can independently increment or decrement its pulse counter value. The remainder of the chapter will mostly focus on channel 0 (ch0) as the functionality of the two channels is identical.

As shown in Figure 11-1, each channel has two input signals:

1. One control signal (e.g. `ctrl_ch0_un`, the control signal for ch0 of unit  $n$ )
2. One input pulse signal (e.g. `sig_ch0_un`, the input pulse signal for ch0 unit  $n$ )

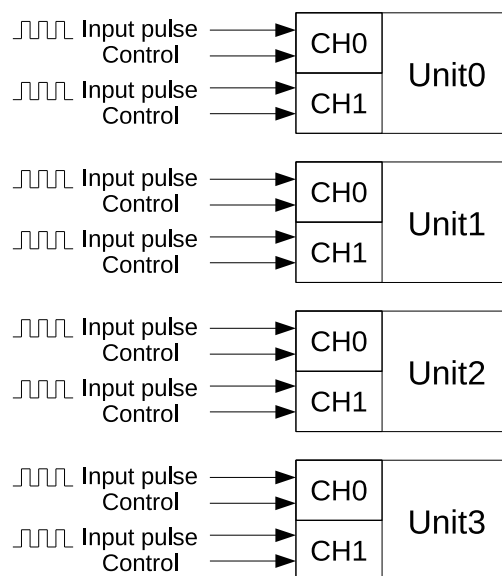


Figure 11-1. PCNT Block Diagram

### 11.1 Features

A PCNT has the following features:

- Four independent pulse counters (units)
- Each unit consists of two independent channels sharing one pulse counter
- All channels have input pulse signals (e.g. `sig_ch0_un`) with their corresponding control signals (e.g. `ctrl_ch0_un`)
- Independent filtering of input pulse signals (`sig_ch0_un` and `sig_ch1_un`) and control signals (`ctrl_ch0_un` and `ctrl_ch1_un`) on each unit
- Each channel has the following parameters:
  1. Selection between counting on positive or negative edges of the input pulse signal

2. Configuration to Increment, Decrement, or Disable counter mode for control signal's high and low states

## 11.2 Functional Description

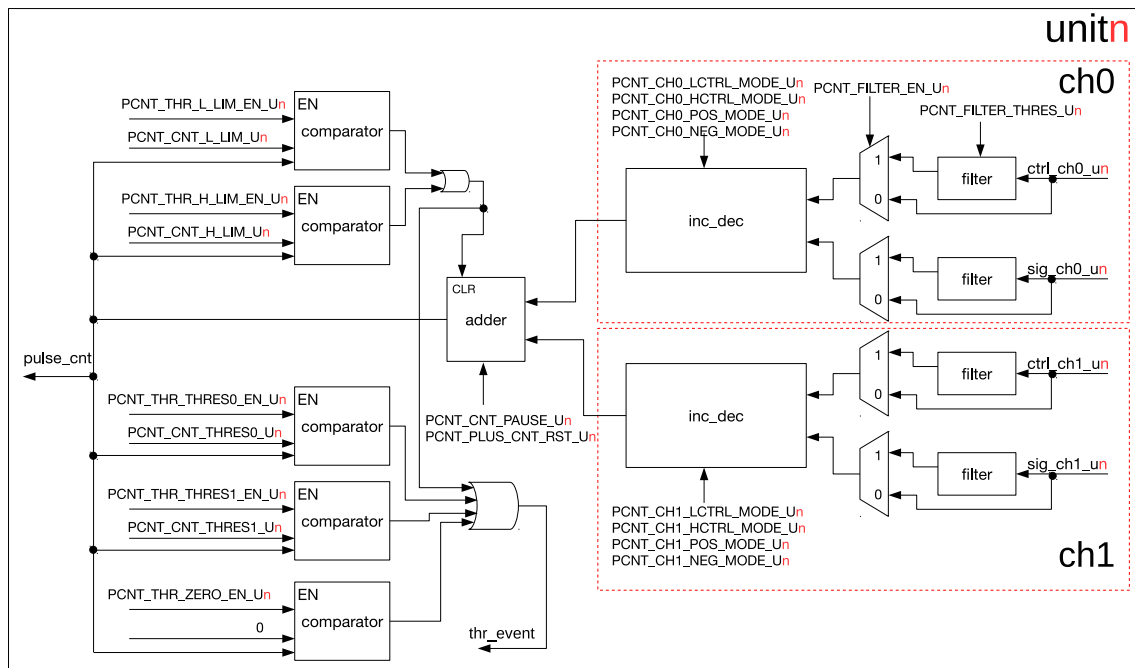


Figure 11-2. PCNT Unit Architecture

Figure 11-2 shows PCNT's architecture. As stated above, `ctrl_ch0_un` is the control signal for ch0 of unit n. Its high and low states can be assigned different counter modes and used for pulse counting of the channel's input pulse signal `sig_ch0_un` on negative or positive edges. The available counter modes are as follows:

- Increment mode: When a channel detects an active edge of `sig_ch0_un` (the one configured for counting), the counter value `pulse_cnt` increases by 1. Upon reaching `PCNT_CNT_H_LIM_Un`, `pulse_cnt` is cleared. If the channel's counter mode is changed or if `PCNT_CNT_PAUSE_Un` is set before `pulse_cnt` reaches `PCNT_CNT_H_LIM_Un`, then `pulse_cnt` freezes and its counter mode changes.
- Decrement mode: When a channel detects an active edge of `sig_ch0_un` (the one configured for counting), the counter value `pulse_cnt` decreases by 1. Upon reaching `PCNT_CNT_L_LIM_Un`, `pulse_cnt` is cleared. If the channel's counter mode is changed or if `PCNT_CNT_PAUSE_Un` is set before `pulse_cnt` reaches `PCNT_CNT_H_LIM_Un`, then `pulse_cnt` freezes and its counter mode changes.
- Disable mode: Counting is disabled, and the counter value `pulse_cnt` freezes.

Table 47 to Table 50 provide information on how to configure the counter mode for channel 0.

Each unit has one filter for all its control and input pulse signals. A filter can be enabled with the bit `PCNT_FILTER_EN_Un`. The filter monitors the signals and ignores all the noise, i.e. the glitches with pulse widths shorter than `PCNT_FILTER_THRES_Un` APB clock cycles in length.

As previously mentioned, each unit has two channels which process different input pulse signals and increase or decrease values via their respective `inc_dec` modules, then the two channels send these values to the adder

**Table 47: Counter Mode. Positive Edge of Input Pulse Signal. Control Signal in Low State**

PCNT_CH0_POS_MODE_U <sub>n</sub>	PCNT_CH0_LCTRL_MODE_U <sub>n</sub>	Counter Mode
1	0	Increment
	1	Decrement
	Others	Disable
2	0	Decrement
	1	Increment
	Others	Disable
Others	N/A	Disable

**Table 48: Counter Mode. Positive Edge of Input Pulse Signal. Control Signal in High State**

PCNT_CH0_POS_MODE_U <sub>n</sub>	PCNT_CH0_HCTRL_MODE_U <sub>n</sub>	Counter Mode
1	0	Increment
	1	Decrement
	Others	Disable
2	0	Decrement
	1	Increment
	Others	Disable
Others	N/A	Disable

**Table 49: Counter Mode. Negative Edge of Input Pulse Signal. Control Signal in Low State**

PCNT_CH0_NEG_MODE_U <sub>n</sub>	PCNT_CH0_LCTRL_MODE_U <sub>n</sub>	Counter Mode
1	0	Increment
	1	Decrement
	Others	Disable
2	0	Decrement
	1	Increment
	Others	Disable
Others	N/A	Disable

**Table 50: Counter Mode. Negative Edge of Input Pulse Signal. Control Signal in High State**

PCNT_CH0_NEG_MODE_U <sub>n</sub>	PCNT_CH0_HCTRL_MODE_U <sub>n</sub>	Counter Mode
1	0	Increment
	1	Decrement
	Others	Disable
2	0	Decrement
	1	Increment
	Others	Disable
Others	N/A	Disable



module that is 16-bit wide with a sign bit. This adder can be suspended by setting `PCNT_CNT_PAUSE_Un`, and cleared by setting `PCNT_PULSE_CNT_RST_Un`.

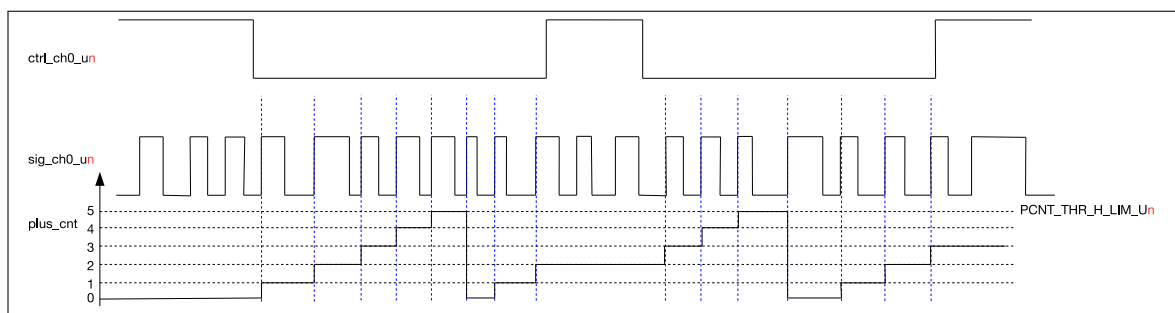
The PCNT has five watchpoints that share one interrupt. The interrupt can be enabled or disabled by interrupt enable signals of each individual watchpoint.

- Maximum count value: When pulse\_cnt reaches `PCNT_CNT_H_LIM_Un`, an interrupt is triggered and `PCNT_CNT_THR_H_LIM_LAT_Un` is high.
- Minimum count value: When pulse\_cnt reaches `PCNT_CNT_L_LIM_Un`, an interrupt is triggered and `PCNT_CNT_THR_L_LIM_LAT_Un` is high.
- Two threshold values: When pulse\_cnt equals either `PCNT_CNT_THRES0_Un` or `PCNT_CNT_THRES1_Un`, an interrupt is triggered and either `PCNT_CNT_THR_THRES0_LAT_Un` or `PCNT_CNT_THR_THRES1_LAT_Un` is high respectively.
- Zero: When pulse\_cnt is 0, an interrupt is triggered and `PCNT_CNT_THR_ZERO_LAT_Un` is valid.

## 11.3 Applications

In each unit, channel 0 and channel 1 can be configured to work independently or together. The three subsections below provide details of channel 0 incrementing independently, channel 0 decrementing independently, and channel 0 and channel 1 incrementing together. For other working modes not elaborated in this section (e.g. channel 1 incrementing/decrementing independently, or one channel incrementing while the other decrementing), reference can be made to these three subsections.

### 11.3.1 Channel 0 Incrementing Independently



**Figure 11-3. Channel 0 Up Counting Diagram**

Figure 11-3 illustrates how channel 0 is configured to increment independently on the positive edge of `sig_ch0_un` while channel 1 is disabled (see subsection 11.2 for how to disable channel 1). The configuration of channel 0 is shown below.

- `PCNT_CH0_LCTRL_MODE_Un=0`: When `ctrl_ch0_un` is low, the counter mode specified for the low state turns on, in this case it is Increment mode.
- `PCNT_CH0_HCTRL_MODE_Un=2`: When `ctrl_ch0_un` is high, the counter mode specified for the low state turns on, in this case it is Disable mode.
- `PCNT_CH0_POS_MODE_Un=1`: The counter increments on the positive edge of `sig_ch0_un`.
- `PCNT_CH0_NEG_MODE_Un=0`: The counter idles on the negative edge of `sig_ch0_un`.
- `PCNT_CNT_H_LIM_Un=5`: When pulse\_cnt counts up to `PCNT_CNT_H_LIM_Un`, it is cleared.

### 11.3.2 Channel 0 Decrementing Independently

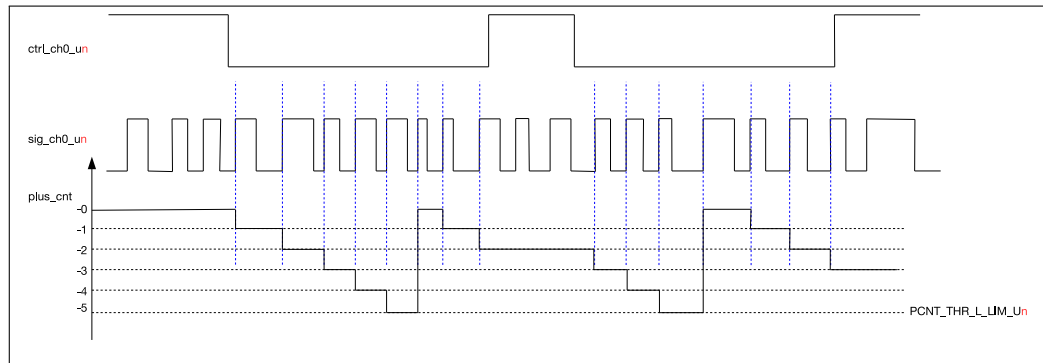


Figure 11-4. Channel 0 Down Counting Diagram

Figure 11-4 illustrates how channel 0 is configured to decrement independently on the positive edge of sig\_ch0\_un while channel 1 is disabled. The configuration of channel 0 in this case differs from that in Figure 11-3 in the following aspects:

- **PCNT\_CH0\_POS\_MODE\_Un=2**: the counter decrements on the positive edge of sig\_ch0\_un.
- **PCNT\_CNT\_L\_LIM\_Un=-5**: when pulse\_cnt counts down to **PCNT\_CNT\_L\_LIM\_Un**, it is cleared.

### 11.3.3 Channel 0 and Channel 1 Incrementing Together

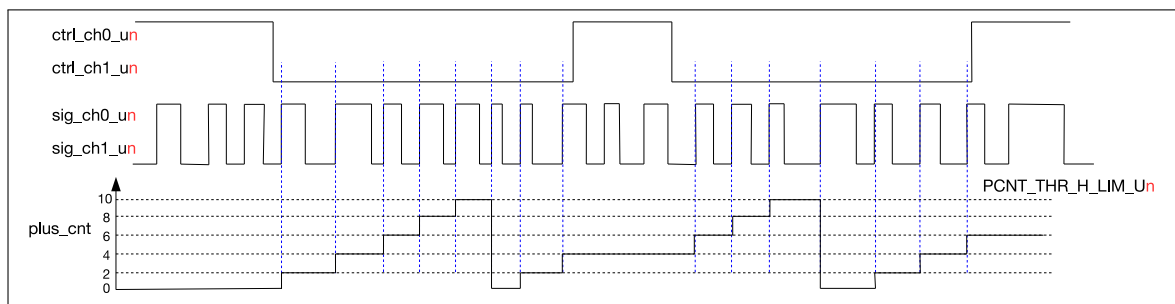


Figure 11-5. Two Channels Up Counting Diagram

Figure 11-5 illustrates how channel 0 and channel 1 are configured to increment on the positive edge of sig\_ch0\_un and sig\_ch1\_un respectively at the same time. It can be seen in Figure 11-5 that control signal ctrl\_ch0\_un and ctrl\_ch1\_un have the same waveform, so as input pulse signal sig\_ch0\_un and sig\_ch1\_un. The configuration procedure is shown below.

- For channel 0:
  - **PCNT\_CH0\_LCTRL\_MODE\_Un=0**: When ctrl\_ch0\_un is low, the counter mode specified for the low state turns on, in this case it is Increment mode.
  - **PCNT\_CH0\_HCTRL\_MODE\_Un=2**: When ctrl\_ch0\_un is high, the counter mode specified for the low state turns on, in this case it is Disable mode.
  - **PCNT\_CH0\_POS\_MODE\_Un=1**: The counter increments on the positive edge of sig\_ch0\_un.
  - **PCNT\_CH0\_NEG\_MODE\_Un=0**: The counter idles on the negative edge of sig\_ch0\_un.
- For channel 1:

- **PCNT\_CH1\_LCTRL\_MODE\_Un=0**: When ctrl\_ch1\_un is low, the counter mode specified for the low state turns on, in this case it is Increment mode.
- **PCNT\_CH1\_HCTRL\_MODE\_Un=2**: When ctrl\_ch1\_un is high, the counter mode specified for the low state turns on, in this case it is Disable mode.
- **PCNT\_CH1\_POS\_MODE\_Un=1**: The counter increments on the positive edge of sig\_ch1\_un.
- **PCNT\_CH1\_NEG\_MODE\_Un=0**: The counter idles on the negative edge of sig\_ch1\_un.
- **PCNT\_CNT\_H\_LIM\_Un=10**: When pulse\_cnt counts up to **PCNT\_CNT\_H\_LIM\_Un**, it is cleared.

## 11.4 Base Address

Users can access the PCNT registers with two base addresses, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter 1 *System and Memory*.

**Table 51: PCNT Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F417000
PeriBUS2	0x60017000

## 11.5 Register Summary

The addresses in the following table are relative to the PCNT base addresses provided in Section 11.4.

Name	Description	Address	Access
<b>Configuration Register</b>			
<a href="#">PCNT_U0_CONF0_REG</a>	Configuration register 0 for unit 0	0x0000	R/W
<a href="#">PCNT_U0_CONF1_REG</a>	Configuration register 1 for unit 0	0x0004	R/W
<a href="#">PCNT_U0_CONF2_REG</a>	Configuration register 2 for unit 0	0x0008	R/W
<a href="#">PCNT_U1_CONF0_REG</a>	Configuration register 0 for unit 1	0x000C	R/W
<a href="#">PCNT_U1_CONF1_REG</a>	Configuration register 1 for unit 1	0x0010	R/W
<a href="#">PCNT_U1_CONF2_REG</a>	Configuration register 2 for unit 1	0x0014	R/W
<a href="#">PCNT_U2_CONF0_REG</a>	Configuration register 0 for unit 2	0x0018	R/W
<a href="#">PCNT_U2_CONF1_REG</a>	Configuration register 1 for unit 2	0x001C	R/W
<a href="#">PCNT_U2_CONF2_REG</a>	Configuration register 2 for unit 2	0x0020	R/W
<a href="#">PCNT_U3_CONF0_REG</a>	Configuration register 0 for unit 3	0x0024	R/W
<a href="#">PCNT_U3_CONF1_REG</a>	Configuration register 1 for unit 3	0x0028	R/W
<a href="#">PCNT_U3_CONF2_REG</a>	Configuration register 2 for unit 3	0x002C	R/W
<a href="#">PCNT_CTRL_REG</a>	Control register for all counters	0x0060	R/W
<b>Status Register</b>			
<a href="#">PCNT_U0_CNT_REG</a>	Counter value for unit 0	0x0030	RO
<a href="#">PCNT_U1_CNT_REG</a>	Counter value for unit 1	0x0034	RO
<a href="#">PCNT_U2_CNT_REG</a>	Counter value for unit 2	0x0038	RO
<a href="#">PCNT_U3_CNT_REG</a>	Counter value for unit 3	0x003C	RO
<a href="#">PCNT_U0_STATUS_REG</a>	PNCT UNIT0 status register	0x0050	RO

Name	Description	Address	Access
<a href="#">PCNT_U1_STATUS_REG</a>	PNCT UNIT1 status register	0x0054	RO
<a href="#">PCNT_U2_STATUS_REG</a>	PNCT UNIT2 status register	0x0058	RO
<a href="#">PCNT_U3_STATUS_REG</a>	PNCT UNIT3 status register	0x005C	RO
<b>Interrupt Register</b>			
<a href="#">PCNT_INT_RAW_REG</a>	Interrupt raw status register	0x0040	RO
<a href="#">PCNT_INT_ST_REG</a>	Interrupt status register	0x0044	RO
<a href="#">PCNT_INT_ENA_REG</a>	Interrupt enable register	0x0048	R/W
<a href="#">PCNT_INT_CLR_REG</a>	Interrupt clear register	0x004C	WO
<b>Version Register</b>			
<a href="#">PCNT_DATE_REG</a>	PCNT version control register	0x00FC	R/W



## 302

ESP32-S2 TRM (Preliminary V0.5)

[Submit Documentation Feedback](#)

**PCNT\_CNT\_THRES1\_U<sub>*n*</sub>** This register is used to configure the thres1 value for unit *n*. (R/W)



Register 11.6: PCNT\_U $n$ \_STATUS\_REG ( $n$ : 0-3) (0x0050+0x4\* $n$ )

(reserved)																								PCNT_CNT_THR_ZERO_LAT_U0 PCNT_CNT_THR_H_LIM_LAT_U0 PCNT_CNT_THR_L_LIM_LAT_U0 PCNT_CNT_THR_THRES0_LAT_U0 PCNT_CNT_THR_THRES1_LAT_U0 PCNT_CNT_THR_ZERO_MODE_U0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
31																								7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**PCNT\_CNT\_THR\_ZERO\_MODE\_U $n$**  The pulse counter status of PCNT\_U $n$  corresponding to 0. 0: pulse counter decreases from positive to 0. 1: pulse counter increases from negative to 0. 2: pulse counter is negative. 3: pulse counter is positive. (RO)

**PCNT\_CNT\_THR\_THRES1\_LAT\_U $n$**  The latched value of thres1 event of PCNT\_U $n$  when threshold event interrupt is valid. 1: the current pulse counter equals to thres1 and thres1 event is valid. 0: others (RO)

**PCNT\_CNT\_THR\_THRES0\_LAT\_U $n$**  The latched value of thres0 event of PCNT\_U $n$  when threshold event interrupt is valid. 1: the current pulse counter equals to thres0 and thres0 event is valid. 0: others (RO)

**PCNT\_CNT\_THR\_L\_LIM\_LAT\_U $n$**  The latched value of low limit event of PCNT\_U $n$  when threshold event interrupt is valid. 1: the current pulse counter equals to thr\_l\_lim and low limit event is valid. 0: others (RO)

**PCNT\_CNT\_THR\_H\_LIM\_LAT\_U $n$**  The latched value of high limit event of PCNT\_U $n$  when threshold event interrupt is valid. 1: the current pulse counter equals to thr\_h\_lim and high limit event is valid. 0: others (RO)

**PCNT\_CNT\_THR\_ZERO\_LAT\_U $n$**  The latched value of zero threshold event of PCNT\_U $n$  when threshold event interrupt is valid. 1: the current pulse counter equals to 0 and zero threshold event is valid. 0: others (RO)



Register 11.7: PCNT\_INT\_RAW\_REG (0x0040)

(reserved)																												PCNT_CNT_THR_EVENT_U3 PCNT_CNT_THR_EVENT_U2 PCNT_CNT_THR_EVENT_U1 PCNT_CNT_THR_EVENT_U0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
31																												4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**PCNT\_CNT\_THR\_EVENT\_U<sub>n</sub>\_INT\_RAW** The raw interrupt status bit for the PCNT\_CNT\_THR\_EVENT\_U<sub>n</sub>\_INT interrupt. (RO)

Register 11.8: PCNT\_INT\_ST\_REG (0x0044)

(reserved)																												PCNT_CNT_THR_EVENT_U3_INT_ST PCNT_CNT_THR_EVENT_U2_INT_ST PCNT_CNT_THR_EVENT_U1_INT_ST PCNT_CNT_THR_EVENT_U0_INT_ST				
31																												4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

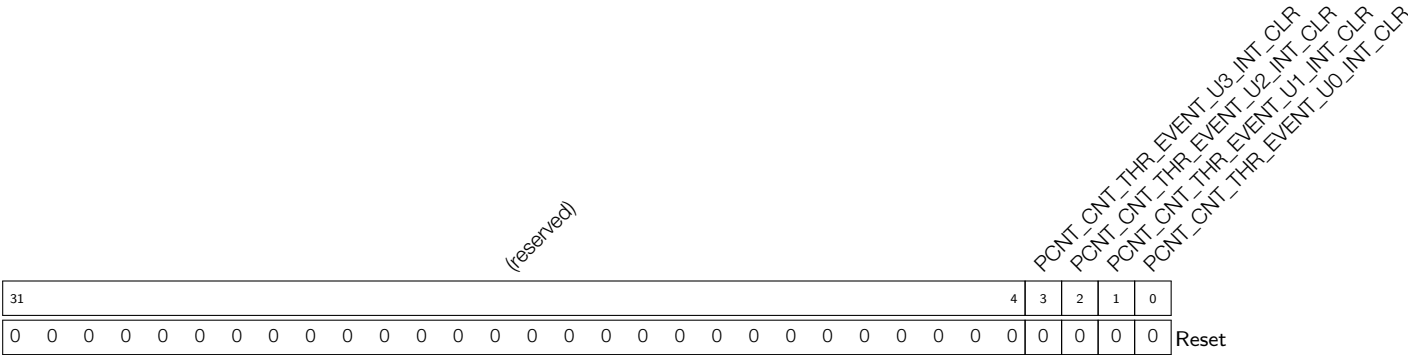
**PCNT\_CNT\_THR\_EVENT\_U<sub>n</sub>\_INT\_ST** The masked interrupt status bit for the PCNT\_CNT\_THR\_EVENT\_U<sub>n</sub>\_INT interrupt. (RO)

Register 11.9: PCNT\_INT\_ENA\_REG (0x0048)

(reserved)																												PCNT_CNT_THR_EVENT_U3_INT_ENA PCNT_CNT_THR_EVENT_U2_INT_ENA PCNT_CNT_THR_EVENT_U1_INT_ENA PCNT_CNT_THR_EVENT_U0_INT_ENA				
31																												4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

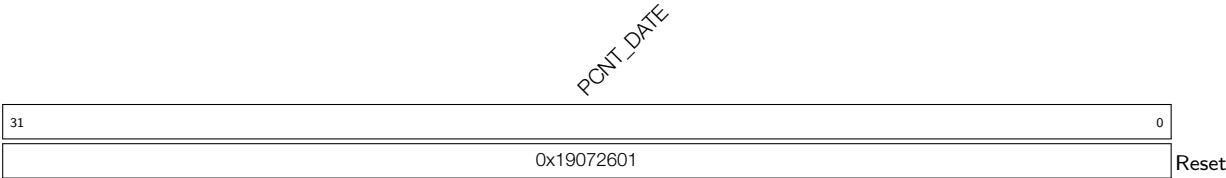
**PCNT\_CNT\_THR\_EVENT\_U<sub>n</sub>\_INT\_ENA** The interrupt enable bit for the PCNT\_CNT\_THR\_EVENT\_U<sub>n</sub>\_INT interrupt. (R/W)

Register 11.10: PCNT\_INT\_CLR\_REG (0x004C)



**PCNT\_CNT\_THR\_EVENT\_U<sub>n</sub>INT\_CLR** Set this bit to clear the PCNT\_CNT\_THR\_EVENT\_U<sub>n</sub>INT interrupt. (WO)

Register 11.11: PCNT\_DATE\_REG (0x00FC)

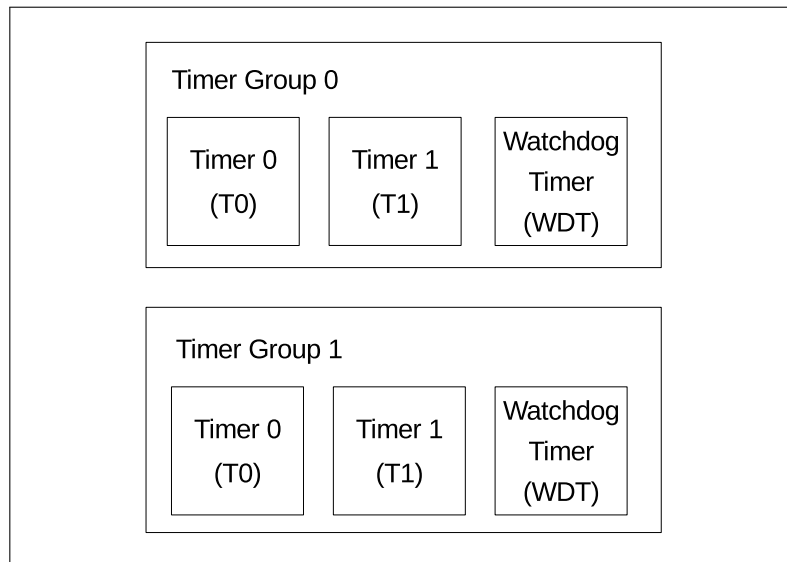


**PCNT\_DATE** This is the PCNT version control register. (R/W)

## 12. 64-bit Timers

### 12.1 Overview

General purpose timers can be used to precisely time an interval, trigger an interrupt after a particular interval (periodically and aperiodically), or act as a hardware clock. As shown in Figure 12-1, the ESP32-S2 chip contains two timer groups, namely timer group 0 and timer group 1. Each timer group consists of two general purpose timers referred to as T<sub>x</sub> (where *x* is 0 or 1) and one Main System Watchdog Timer. All general purpose timers are based on 16-bit prescalers and 64-bit auto-reload-capable up/down counters.



**Figure 12-1. Timer Units within Groups**

Note that while the Main System Watchdog Timer registers are described in this chapter, their functional description is included in the Chapter 13: *Watchdog Timers*. Therefore, the term ‘timers’ within this chapter refers to the general purpose timers.

The timers’ features are summarized as follows:

- A 16-bit clock prescaler, from 1 to 65536
- A 64-bit time-base counter programmable to be incrementing or decrementing
- Able to read real-time value of the time-base counter
- Halting and resuming the time-base counter
- Programmable alarm generation
- Timer value reload (Auto-reload at alarm or software-controlled instant reload)
- Level and edge interrupt generation

## 12.2 Functional Description

### 12.2.1 16-bit Prescaler and Clock Selection

Each timer can select between the APB clock (APB\_CLK) or external clock (XTAL\_CLK) as its clock source by setting the [TIMG\\_Tx\\_USE\\_XTAL](#) field of the [TIMG\\_TxCONFIG\\_REG](#) register. The clock is then divided by a 16-bit prescaler to generate the time-base counter clock (TB\_CLK) used by the time-base counter. The 16-bit prescaler is configured by the [TIMG\\_Tx\\_DIVIDER](#) field and can take any value from 1 to 65536. Note that programming a value of 0 in [TIMG\\_Tx\\_DIVIDER](#) will result in the divisor being 65536.

The timer must be disabled (i.e. [TIMG\\_Tx\\_EN](#) should be cleared) before modifying the 16-bit prescaler. Modifying the 16-bit prescaler whilst the timer is enabled can lead to unpredictable results.

### 12.2.2 64-bit Time-based Counter

The 64-bit time-base counters are based on TB\_CLK and can be configured to increment or decrement via the [TIMG\\_Tx\\_INCREASE](#) field. The time-base counter can be enabled/disabled by setting/clearing the [TIMG\\_Tx\\_EN](#) field. Whilst enabled, the time-base counter will increment/decrement on each cycle of TB\_CLK. When disabled, the time-base counter is essentially frozen. Note that the [TIMG\\_Tx\\_INCREASE](#) field can be changed whilst [TIMG\\_Tx\\_EN](#) is set and will cause the time-base counter to change direction instantly.

To read the 64-bit current timer value of the time-base counter, the timer value must be latched to two registers before being read by the CPU (due to the CPU being 32-bit). By writing any value to the [TIMG\\_TxUPDATE\\_REG](#), the current value of the 64-bit timer is instantly latched into the [TIMG\\_TxLO\\_REG](#) and [TIMG\\_TxHI\\_REG](#) registers containing the lower and upper 32-bits respectively. [TIMG\\_TxLO\\_REG](#) and [TIMG\\_TxHI\\_REG](#) registers will remain unchanged for the CPU to read in its own time until [TIMG\\_TxUPDATE\\_REG](#) is written to again.

### 12.2.3 Alarm Generation

A timer can be configured to trigger an alarm when the timer's current value matches the alarm value. An alarm will cause an interrupt to occur and (optionally) an automatic reload of the timer's current value (see Section 12.2.4). The 64-bit alarm value is configured in the [TIMG\\_TxALARMLO\\_REG](#) and [TIMG\\_TxALARMHI\\_REG](#) representing the lower and upper 32-bits of the alarm value respectively. However, the configured alarm value is ineffective until the alarm is enabled by setting the [TIMG\\_Tx\\_ALARM\\_EN](#) field. In order to simplify the scenario where the alarm is enabled 'too late' (i.e. the timer value has already passed the alarm value when the alarm is enabled), the alarm value will also trigger immediately if the current timer value is larger/smaller than the alarm value for an up-counting/down-counting timer.

When an alarm occurs, the [TIMG\\_Tx\\_ALARM\\_EN](#) field is automatically cleared and no alarm will occur again until the [TIMG\\_Tx\\_ALARM\\_EN](#) is set.

### 12.2.4 Timer Reload

A timer is reloaded when a timer's current value is overwritten with a reload value stored in the [TIMG\\_Tx\\_LOAD\\_LO](#) and [TIMG\\_Tx\\_LOAD\\_HI](#) registers that correspond to the lower and upper 32-bits of the timer's new value respectively. However, writing a reload value to [TIMG\\_Tx\\_LOAD\\_LO](#) and [TIMG\\_Tx\\_LOAD\\_HI](#) will not cause the timer's current value to change. Instead, the reload value is ignored by the timer until a reload event occurs. A reload event can be triggered either by a software instant reload or an auto-reload at alarm.

A software instant reload is triggered by the CPU writing any value to [TIMG\\_TxLOAD\\_REG](#) causing the timer's current value to be instantly reloaded. If [TIMG\\_Tx\\_EN](#) is set, the timer will continue incrementing/decrementing from the new value. If [TIMG\\_Tx\\_EN](#) is cleared, the timer will remain frozen at the new value until counting is

re-enabled.

An auto-reload at alarm will cause a timer reload when an alarm occurs thus allowing the timer to continue incrementing/decrementing from the reload value. This is generally useful for resetting the timer's value when using periodic alarms. To enable auto-reload at alarm, the `TIMG_Tx_AUTORELOAD` field should be set. If not enabled, the timer's value will continue to increment/decrement past the alarm value after an alarm.

### 12.2.5 Interrupts

Each timer has its own pair of interrupt lines (for edge and level interrupts) that can be routed to the CPU. Thus, there are a total of six interrupt lines per timer group and they are named as follows:

- `TIMG_WDT_LEVEL_INT`: Level interrupt line for the watchdog timer in the group, generated when a watchdog timer interrupt stage times out.
- `TIMG_WDT_EDGE_INT`: Edge interrupt line for the watchdog timer in the group, generated when a watchdog timer interrupt stage times out.
- `TIMG_Tx_LEVEL_INT`: Level interrupt for one of the general purpose timers, generated when an alarm event happens.
- `TIMG_Tx_EDGE_INT`: Edge interrupt for one of the general purpose timer, generated when an alarm event happens.

Interrupts are triggered after an alarm (or stage timeout for watchdog timers) occurs. Level interrupt lines will be held high after an alarm (or stage timeout) occurs, and will remain so until manually cleared. Conversely, edge interrupts will generate a short pulse after an alarm (or stage timeout) occurs. To enable a timer's level or edge interrupt lines, the `TIMG_Tx_LEVEL_INT_EN` or `TIMG_Tx_EDGE_INT_EN` bits should be set respectively.

The interrupts of each timer group are governed by a set of registers. Each timer within the group will have a corresponding bit in each of these registers:

- `TIMG_Tx_INT_RAW` : An alarm event sets it to 1. The bit will remain set until writing to the timer's corresponding bit in `TIMG_Tx_INT_CLR`.
- `TIMG_WDT_INT_RAW` : A stage time out will set the timer's bit to 1. The bit will remain set until writing to the timer's corresponding bit in `TIMG_WDT_INT_CLR`.
- `TIMG_Tx_INT_ST` : Reflects the status of each timer's interrupt and is generated by masking the bits of `TIMG_Tx_INT_RAW` with `TIMG_Tx_INT_ENA`. For level interrupts, these bits reflect the level on the watchdog timer's level interrupt line.
- `TIMG_WDT_INT_ST` : Reflects the status of each watchdog timer's interrupt and is generated by masking the bits of `TIMG_WDT_INT_RAW` with `TIMG_WDT_INT_ENA`. For level interrupts, these bits reflect the level on the watchdog timer's level interrupt line.
- `TIMG_Tx_INT_ENA` : Used to enable or mask the interrupt status bits of timers within the group.
- `TIMG_WDT_INT_ENA` : Used to enable or mask the interrupt status bits of watchdog timer within the group.
- `TIMG_Tx_INT_CLR` : Used to clear a timer's interrupt by setting its corresponding bit to 1. The timer's corresponding bit in `TIMG_Tx_INT_RAW` and `TIMG_Tx_INT_ST` will be cleared as a result. Note that a timer's interrupt must be cleared before the next interrupt occurs when using level interrupts.
- `TIMG_WDT_INT_CLR` : Used to clear a timer's interrupt by setting its corresponding bit to 1. The watchdog timer's corresponding bit in `TIMG_WDT_INT_RAW` and `TIMG_WDT_INT_ST` will be cleared as a result.

Note that a watchdog timer's interrupt must be cleared before the next interrupt occurs when using level interrupts.

## 12.3 Configuration and Usage

### 12.3.1 Timer as a Simple Clock

1. Configure the time-base counter
  - Select clock source by setting `TIMG_Tx_USE_XTAL` field.
  - Configure the 16-bit prescaler by setting `TIMG_Tx_DIVIDER`.
  - Configure the timer direction by setting/clearing `TIMG_Tx_INCREASE`.
  - Set the timer's starting value by writing the starting value to `TIMG_Tx_LOAD_LO` and `TIMG_Tx_LOAD_HI`, then reloading it into the timer by writing any value to `TIMG_TxLOAD_REG`.
2. Start the timer by setting `TIMG_Tx_EN`.
3. Getting the timer's current value.
  - Write any value to `TIMG_TxUPDATE_REG` to latch the timer's current value.
  - Read the latched timer value from `TIMG_TxLO_REG` and `TIMG_TxHI_REG`.

### 12.3.2 Timer as One-shot Alarm

1. Configure the time-base counter following step 1 of Section 12.3.1.
2. Configure the alarm.
  - Configure the alarm value by setting `TIMG_TxALARMLO_REG` and `TIMG_TxALARMHI_REG`.
  - Enable interrupt by setting `TIMG_Tx_LEVEL_INT_EN` or `TIMG_Tx_EDGE_INT_EN` for level or edge interrupts respectively.
3. Disable auto reload by clearing `TIMG_Tx_AUTORELOAD`.
4. Start the timer by setting `TIMG_Tx_EN`.
5. Handle the alarm interrupt.
  - Clear the interrupt by setting the timer's corresponding bit in `TIMG_Tx_INT_CLR`.
  - Disable the timer by clearing `TIMG_Tx_EN`.

### 12.3.3 Timer as Periodic Alarm

1. Configure the time-base counter following step 1 of Section 12.3.1.
2. Configure the alarm following step 2 of Section 12.3.2.
3. Enable auto reload by setting `TIMG_Tx_AUTORELOAD` and setting the reload value in `TIMG_Tx_LOAD_LO` and `TIMG_Tx_LOAD_HI`.
4. Start the timer by setting `TIMG_Tx_EN`.
5. Handle the alarm interrupt (repeat on each alarm iteration).
  - Clear the interrupt by setting the timer's corresponding bit in `TIMG_Tx_INT_CLR`.
  - If the next alarm requires a new alarm value and reload value (i.e. different alarm interval per iteration), then `TIMG_Tx_ALARMLO_REG`, `TIMG_Tx_ALARMHI_REG`, `TIMG_Tx_LOAD_LO`, and `TIMG_Tx_LOAD_HI` should be reconfigured as needed. Otherwise, the aforementioned registers should remain unchanged.
  - Re-enable the alarm by setting `TIMG_Tx_ALARM_EN`.
6. Stopping the timer (on final alarm iteration).
  - Clear the interrupt by setting the timer's corresponding bit in `TIMG_Tx_INT_CLR`.
  - Disable the timer by clearing `TIMG_Tx_EN`.

## 12.4 Base Address

Users can access the 64-bit Timer with four base addresses, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter 1 *System and Memory*.

**Table 53: 64-bit Timers Base Address**

Module	Bus to Access Peripheral	Base Address
TIMG0	PeriBUS1	0x3F41F000
	PeriBUS2	0x6001F000
TIMG1	PeriBUS1	0x3F420000
	PeriBUS2	0x60020000

## 12.5 Register Summary

The addresses in the following table are relative to the 64-bit Timer base addresses provided in Section 12.4.

Name	Description	Address	Access
<b>Configuration and Control Register</b>			
<code>TIMG_T0CONFIG_REG</code>	Timer 0 configuration register	0x0000	R/W
<code>TIMG_T0LO_REG</code>	Timer 0 current value, low 32 bits	0x0004	RO
<code>TIMG_T0HI_REG</code>	Timer 0 current value, high 32 bits	0x0008	RO
<code>TIMG_T0UPDATE_REG</code>	Write to copy current timer value to <code>TIMGn_T0_(LO/HI)_REG</code>	0x000C	R/W
<code>TIMG_T0ALARMLO_REG</code>	Timer 0 alarm value, low 32 bits	0x0010	R/W

Name	Description	Address	Access
<a href="#">TIMG_T0ALARMHI_REG</a>	Timer 0 alarm value, high bits	0x0014	R/W
<a href="#">TIMG_T0LOADLO_REG</a>	Timer 0 reload value, low 32 bits	0x0018	R/W
<a href="#">TIMG_T0LOADHI_REG</a>	Timer 0 reload value, high 32 bits	0x001C	R/W
<a href="#">TIMG_T0LOAD_REG</a>	Write to reload timer from TIMG_T0_(LOADLOLOADHI)_REG	0x0020	WO
<a href="#">TIMG_T1CONFIG_REG</a>	Timer 1 configuration register	0x0024	R/W
<a href="#">TIMG_T1LO_REG</a>	Timer 1 current value, low 32 bits	0x0028	RO
<a href="#">TIMG_T1HI_REG</a>	Timer 1 current value, high 32 bits	0x002C	RO
<a href="#">TIMG_T1UPDATE_REG</a>	Write to copy current timer value to TIMGn_T1_(LO/HI)_REG	0x0030	R/W
<a href="#">TIMG_T1ALARMLO_REG</a>	Timer 1 alarm value, low 32 bits	0x0034	R/W
<a href="#">TIMG_T1ALARMHI_REG</a>	Timer 1 alarm value, high bits	0x0038	R/W
<a href="#">TIMG_T1LOADLO_REG</a>	Timer 1 reload value, low 32 bits	0x003C	R/W
<a href="#">TIMG_T1LOADHI_REG</a>	Timer 1 reload value, high 32 bits	0x0040	R/W
<a href="#">TIMG_T1LOAD_REG</a>	Write to reload timer from TIMG_T0_(LOADLOLOADHI)_REG	0x0044	WO
<b>Configuration and Control Register for WDT</b>			
<a href="#">TIMG_WDTCONFIG0_REG</a>	Watchdog timer configuration register	0x0048	R/W
<a href="#">TIMG_WDTCONFIG1_REG</a>	Watchdog timer prescaler register	0x004C	R/W
<a href="#">TIMG_WDTCONFIG2_REG</a>	Watchdog timer stage 0 timeout value	0x0050	R/W
<a href="#">TIMG_WDTCONFIG3_REG</a>	Watchdog timer stage 1 timeout value	0x0054	R/W
<a href="#">TIMG_WDTCONFIG4_REG</a>	Watchdog timer stage 2 timeout value	0x0058	R/W
<a href="#">TIMG_WDTCONFIG5_REG</a>	Watchdog timer stage 3 timeout value	0x005C	R/W
<a href="#">TIMG_WDTFEED_REG</a>	Write to feed the watchdog timer	0x0060	WO
<a href="#">TIMG_WDTWPROTECT_REG</a>	Watchdog write protect register	0x0064	R/W
<b>Configuration and Control Register for RTC CALI</b>			
<a href="#">TIMG_RTCCALICFG2_REG</a>	Timer group calibration register	0x00A8	varies
<b>Interrupt Register</b>			
<a href="#">TIMG_INT_ENA_TIMERS_REG</a>	Interrupt enable bits	0x0098	R/W
<a href="#">TIMG_INT_RAW_TIMERS_REG</a>	Raw interrupt status	0x009C	RO
<a href="#">TIMG_INT_ST_TIMERS_REG</a>	Masked interrupt status	0x00A0	RO
<a href="#">TIMG_INT_CLR_TIMERS_REG</a>	Interrupt clear bits	0x00A4	WO
<b>Version Register</b>			
<a href="#">TIMG_TIMERS_DATE_REG</a>	Version control register	0x00F8	R/W
<b>Configuration Register</b>			
<a href="#">TIMG_REGCLK_REG</a>	Timer group clock gate register	0x00FC	R/W





[Submit Documentation Feedback](#)



of timer **x** can be read here. (RO)

TIMG\_Tx\_UPDATE



**TIMG\_Tx\_UPDATE** After writing 0 or 1 to TIMG\_TxUPDATE\_REG, the counter value is latched. (R/W)

TIMG\_T<sub>x</sub>\_ALARM\_LO

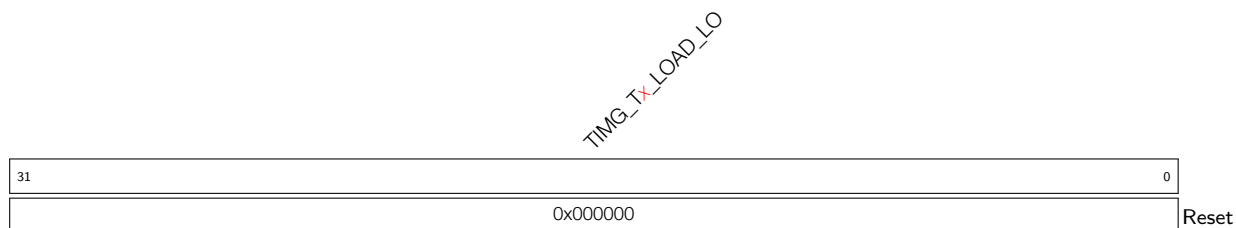


**TIMG\_T<sub>x</sub>\_ALARM\_LO** Timer *x* alarm trigger time-base counter value, low 32 bits. (R/W)

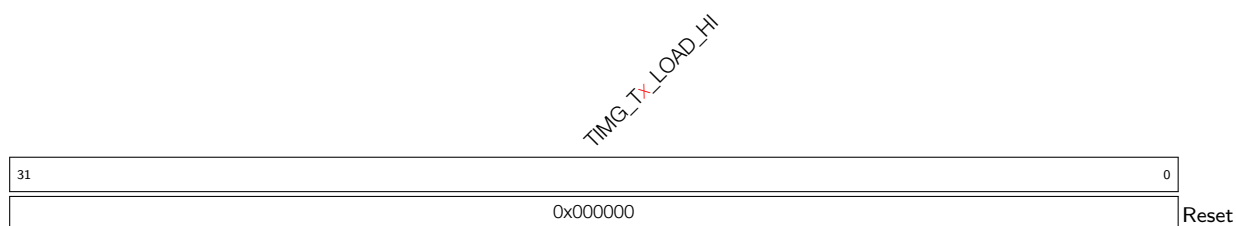
TIMG\_T~~x~~\_ALARM\_HI



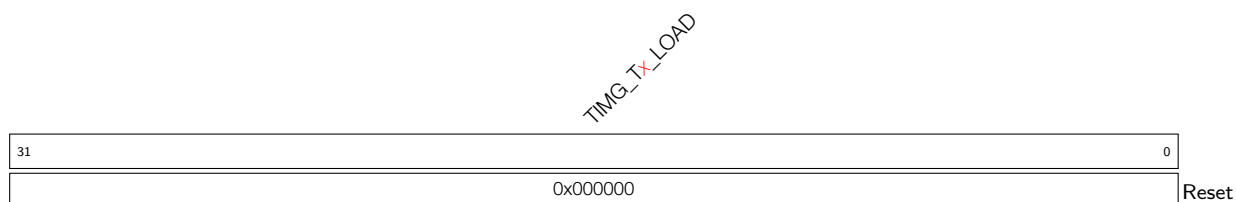
**TIMG\_T<sub>x</sub>\_ALARM\_HI** Timer *x* alarm trigger time-base counter value, high 32 bits. (R/W)

**Register 12.7: TIMG\_T<sub>x</sub>LOADLO\_REG (x: 0-1) (0x0018+0x24\*x)**

**TIMG\_T<sub>x</sub>LOAD\_LO** Low 32 bits of the value that a reload will load onto timer *x* time-base Counter. (R/W)

**Register 12.8: TIMG\_T<sub>x</sub>LOADHI\_REG (x: 0-1) (0x001C+0x24\*x)**

**TIMG\_T<sub>x</sub>LOAD\_HI** High 32 bits of the value that a reload will load onto timer *x* time-base counter. (R/W)

**Register 12.9: TIMG\_T<sub>x</sub>LOAD\_REG (x: 0-1) (0x0020+0x24\*x)**

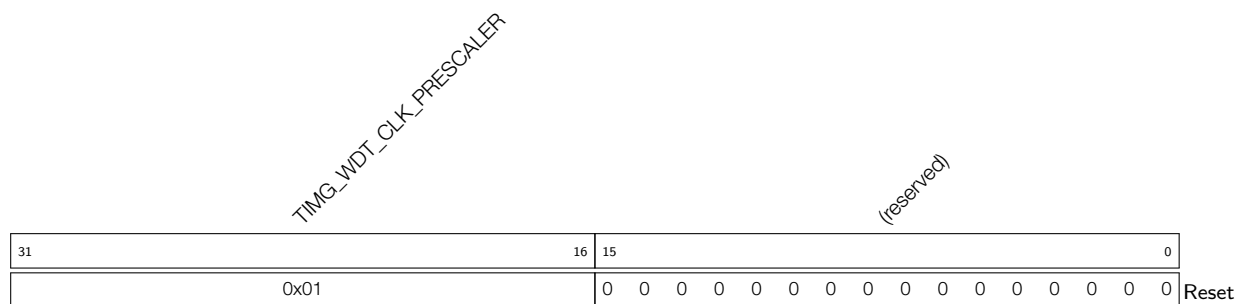
**TIMG\_T<sub>x</sub>LOAD** Write any value to trigger a timer *x* time-base counter reload. (WO)

## 316

ESP32-S2 TRM (Preliminary V0.5)[Submit Documentation Feedback](#)

**TIMG\_WDT\_EN** When set, MWDT is enabled. (R/W)

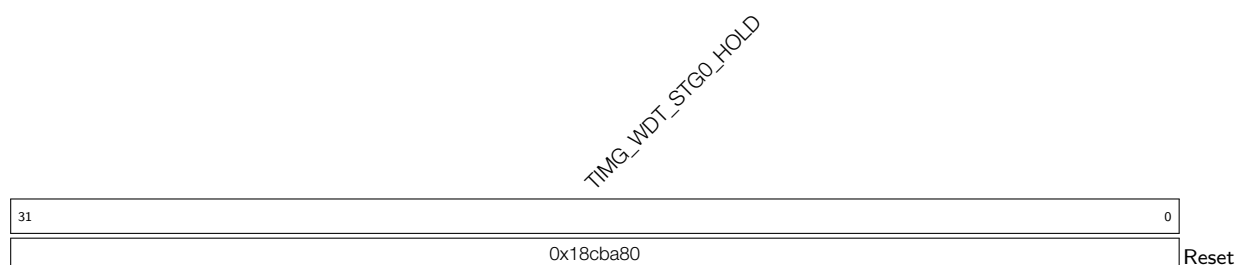
Register 12.11: TIMG\_WDTCONFIG1\_REG (0x004C)



**TIMG\_WDT\_CLK\_PRESCALER** MWDT clock prescaler value. MWDT clock period = 12.5 ns \*

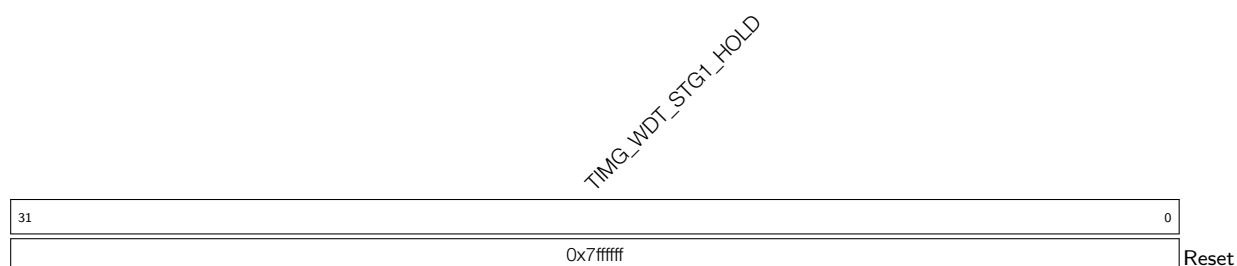
TIMG\_WDT\_CLK\_PRESCALE. (R/W)

Register 12.12: TIMG\_WDTCONFIG2\_REG (0x0050)



**TIMG\_WDT\_STG0\_HOLD** Stage 0 timeout value, in MWDT clock cycles. (R/W)

Register 12.13: TIMG\_WDTCONFIG3\_REG (0x0054)



**TIMG\_WDT\_STG1\_HOLD** Stage 1 timeout value, in MWDT clock cycles. (R/W)

**Register 12.14: TIMG\_WDTCONFIG4\_REG (0x0058)**

TIMG\_WDT\_STG2\_HOLD

31	0
0x0ffff	
Reset	

**TIMG\_WDT\_STG2\_HOLD** Stage 2 timeout value, in MWDT clock cycles. (R/W)

**Register 12.15: TIMG\_WDTCONFIG5\_REG (0x005C)**

TIMG\_WDT\_STG3\_HOLD

31	0
0x0ffff	
Reset	

**TIMG\_WDT\_STG3\_HOLD** Stage 3 timeout value, in MWDT clock cycles. (R/W)

**Register 12.16: TIMG\_WDTFEED\_REG (0x0060)**

TIMG\_WDT\_FEED

31	0
0x000000	
Reset	

**TIMG\_WDT\_FEED** Write any value to feed the MWDT. (WO) (WO)

**Register 12.17: TIMG\_WDTWPROTECT\_REG (0x0064)**

TIMG\_WDT\_WKEY

31	0
0x50d83aa1	
Reset	

**TIMG\_WDT\_WKEY** If the register contains a different value than its reset value, write protection is enabled. (R/W)



Register 12.21: TIMG\_INT\_ST\_TIMERS\_REG (0x00A0)

(reserved)																																TIMG_WDT_INT_ST TIMG_T1_INT_ST TIMG_TO_INT_ST																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
31																															3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**TIMG\_T<sub>x</sub>\_INT\_ST** The masked interrupt status bit for the TIMG\_T<sub>x</sub>\_INT interrupt. (RO)

**TIMG\_WDT\_INT\_ST** The masked interrupt status bit for the TIMG\_WDT\_INT interrupt. (RO)

Register 12.22: TIMG\_INT\_CLR\_TIMERS\_REG (0x00A4)

(reserved)																															TIMG_WDT_INT_CLR TIMG_T1_INT_CLR TIMG_TO_INT_CLR																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
31																															3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**TIMG\_T<sub>x</sub>\_INT\_CLR** Set this bit to clear the TIMG\_T<sub>x</sub>\_INT interrupt. (WO)

**TIMG\_WDT\_INT\_CLR** Set this bit to clear the TIMG\_WDT\_INT interrupt. (WO)

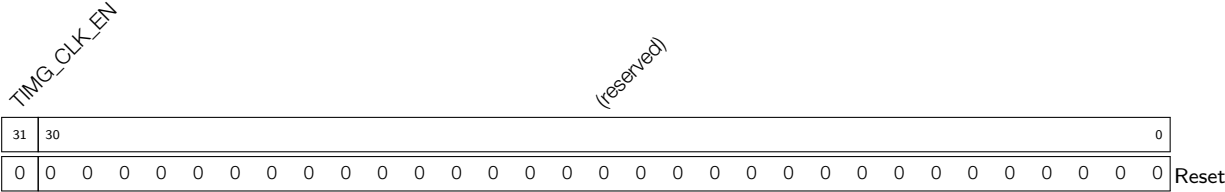
Register 12.23: TIMG\_TIMERS\_DATE\_REG (0x00F8)

(reserved)				TIMG_TIMERS_DATE																												
31				28																								27				0
0	0	0	0	0x1907261																												Reset

**TIMG\_TIMERS\_DATE** Version control register. (R/W)



Register 12.24: TIMG\_REGCLK\_REG (0x00FC)



**TIMG\_CLK\_EN** Register clock gate signal. 1: Registers can be read and written to by software. 0: Registers can not be read or written to by software. (R/W)

## 13. Watchdog Timers

### 13.1 Overview

Watchdog timers are hardware timers used to detect and recover from malfunctions. They must be periodically fed (reset) to prevent a timeout. A system/software that is behaving unexpectedly (e.g. is stuck in a software loop or in overdue events) will fail to feed the watchdog thus trigger a watchdog time out. Therefore, watchdog timers are useful for detecting and handling erroneous system/software behavior.

The ESP32-S2 contains three watchdog timers: one in each of the two timer groups (called Main System Watchdog Timers, or MWDT) and one in the RTC Module (called the RTC Watchdog Timer, or RWDT). Each watchdog timer allows for four separately configurable stages and each stage can be programmed to take one of three (or four for RWDT) actions upon expiry, unless the watchdog is fed or disabled. The actions upon expiry are: interrupt, CPU reset, core reset and system reset. Only RWDT can trigger a system reset that will reset the entire digital circuits, which is the main system including the RTC itself. A timeout value can be set for each stage individually.

During the flash boot process, RWDT and the first MWDT are enabled automatically in order to detect and recover from booting errors.

Note that while this chapter provides the functional descriptions of the watchdog timer's, their register descriptions are provided in Chapter 12: [64-bit Timers](#).

### 13.2 Features

Watchdog timers have the following features:

- Four stages, each with a programmable timeout value. Each stage can be configured and enabled/disabled separately
- One of three/four (for MWDTs/ RWDT) possible actions (interrupt, CPU reset, core reset and system reset) available upon expiry of each stage
- 32-bit expiry counter
- Write protection, to prevent RWDT and MWDT configuration from being altered inadvertently
- Flash boot protection  
If the boot process from an SPI flash does not complete within a predetermined period of time, the watchdog will reboot the entire main system.

### 13.3 Functional Description

#### 13.3.1 Clock Source and 32-Bit Counter

At the core of each watchdog timer is a 32-bit counter. The clock source of MWDTs is derived from the APB clock via a pre-MWDT 16-bit configurable prescaler. Conversely, the clock source of RWDT is derived directly from a RTC slow clock (without a prescaler) which is usually running at 32 kHz. The 16-bit prescaler for MWDTs is configured via the `TIMG_WDT_CLK_PRESCALER` field of `TIMG_WDTCONFIG1_REG`.

MWDTs and RWDT are enabled by setting the `TIMG_WDT_EN` and `RTC_CNTL_WDT_EN` fields respectively. When enabled, the 32-bit counters of each watchdog will increment on each source clock cycle until the timeout value of the current stage is reached (i.e. expiry of the current stage). When this occurs, the current counter value

is reset to zero and the next stage will become active. If a watchdog timer is fed by software, the timer will return to stage 0 and reset its counter value to zero. Software can feed a watchdog timer by writing any value to [TIMG\\_WDTFEED\\_REG](#) for MDWTs and [RTC\\_CNTL\\_RTC\\_WDT\\_FEED](#) for RWDT.

### 13.3.2 Stages and Timeout Actions

Timer stages allow for a timer to have a series of different timeout values and corresponding expiry action. When one stage expires, the expiry action is triggered, the counter value is reset to zero, and the next stage becomes active. MDWTs/ RWDT provide four stages (called stages 0 to 3). The watchdog timers will progress through each stage in a loop (i.e. from stage 0 to 3, then back to stage 0).

Timeout values of each stage for MDWTs are configured in [TIMG\\_WDTCONFIG<sub>i</sub>\\_REG](#) (where *i* ranges from 2 to 5), whilst timeout values for RWDT are configured using [RTC\\_CNTL\\_WDT\\_STG<sub>j</sub>\\_HOLD](#) field (where *j* ranges from 0 to 3).

Please note that the timeout value of stage 0 for RWDT ( $T_{hold0}$ ) is determined together by [EFUSE\\_WDT\\_DELAY\\_SEL](#) field of an eFuse register and [RTC\\_CNTL\\_WDT\\_STG0\\_HOLD](#). The relationship is as follows:

$$T_{hold0} = \text{RTC\_CNTL\_WDT\_STG0\_HOLD} \ll \text{EFUSE\_WDT\_DELAY\_SEL} + 1$$

Upon the expiry of each stage, one of the following expiry actions will be executed:

- Trigger an interrupt  
When the stage expires, an interrupt is triggered.
- Reset a CPU core  
When the stage expires, the CPU core will be reset.
- Reset the main system  
When the stage expires, the main system (which includes MDWTs, CPU, and all peripherals) will be reset. The RTC will not be reset.
- Reset the main system and RTC  
When the stage expires the main system and the RTC will both be reset. This action is only available in RWDT.
- Disabled  
This stage will have no effects on the system.

For MDWTs, the expiry action of all stages is configured in [TIMG\\_WDTCONFIG0\\_REG](#). Likewise for RWDT, the expiry action is configured in [RTC\\_CNTL\\_WDTCONFIG0\\_REG](#).

### 13.3.3 Write Protection

Watchdog timers are critical to detecting and handling erroneous system/software behavior, thus should not be disabled easily (e.g. due to a misplaced register write). Therefore, MDWTs and RWDT incorporate a write protection mechanism that prevent the watchdogs from being disabled or tampered with due to an accidental write. The write protection mechanism is implemented using a write-key register for each timer ([TIMG\\_WDT\\_WKEY](#) for MDWT, [RTC\\_CNTL\\_WDT\\_WKEY](#) for RWDT). The value 0x50D83AA1 must be written to the watchdog timer's write-key register before any other register of the same watchdog timer can be changed. Any attempts to write to a watchdog timer's registers (other than the write-key register itself) whilst the write-key register's value is not 0x50D83AA1 will be ignored. The recommended procedure for accessing a watchdog timer is as follows:

1. Disable the write protection by writing the value 0x50D83AA1 to the timer's write-key register.
2. Make the required modification of the watchdog such as feeding or changing its configuration.
3. Re-enable write protection by writing any value other than 0x50D83AA1 to the timer's write-key register.

### 13.3.4 Flash Boot Protection

During flash booting process, MWDT in timer group 0 ([TIMGO](#)), as well as RWDT, are automatically enabled.

Stage 0 for the enabled MWDT is automatically configured to reset the system upon expiry. Likewise, stage 0 for RWDT is configured to reset the main system and RTC when it expires. After booting,

[TIMG\\_WDT\\_FLASHBOOT\\_MOD\\_EN](#) and [RTC\\_CNTL\\_WDT\\_FLASHBOOT\\_MOD\\_EN](#) should be cleared to stop the flash boot protection procedure for both MWDT and RWDT respectively. After this, MWDT and RWDT can be configured by software.

## 13.4 Super Watchdog

Super watchdog (SWD) is an ultra-low-power circuit that helps to prevent the system from operating in a sub-optimal state and resets the system if required. SWD contains a watchdog circuit that needs to be fed for nearly every 1 s. About 100 ms before watchdog timeout, it will also send out a WD\_INTR signal as a request to remind the system to feed the watchdog.

If the system doesn't respond to SWD feed request and watchdog finally times out, SWD will generate a system level signal SWD\_RSTB to reset the whole digital circuits on chip.

### 13.4.1 Features

SWD has the following features:

- Ultra-low power and small in area
- Interrupt to remind the system to feed SWD
- Various dedicated methods for software to feed SWD, which enables SWD to monitor the working state of the whole operating system

### 13.4.2 Super Watchdog Controller

### 13.4.2.1 Structure

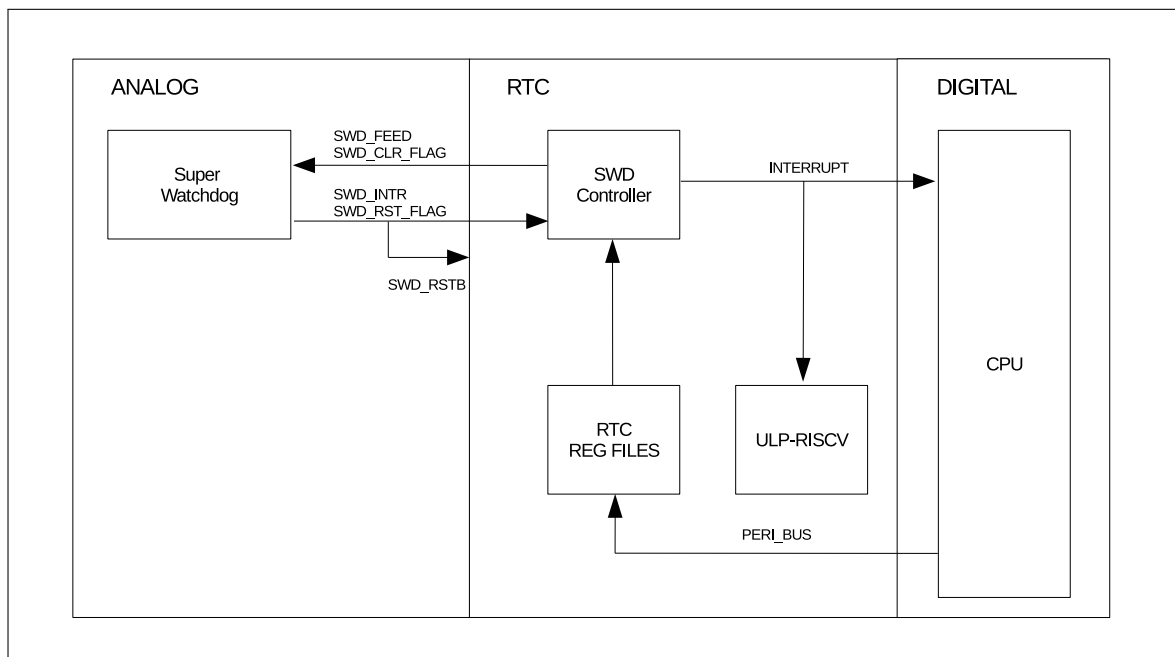


Figure 13-1. Super Watchdog Controller Structure

### 13.4.2.2 Workflow

In normal state:

- SWD controller receives feed request from SWD.
- SWD controller can send an interrupt to main CPU or ULP-RISC-V.
- Main CPU can decide whether to feed SWD directly by setting [RTC\\_CNTL\\_SWD\\_FEED](#), or send an interrupt to ULP-RISC-V and ask ULP-RISC-V to feed SWD by setting [RTC\\_CNTL\\_SWD\\_FEED](#).
- When trying to feed SWD, CPU or the co-processor needs to disable SWD controller's write protection by writing 0x8F1D312A to [RTC\\_CNTL\\_SWD\\_WKEY](#). This prevents SWD from being fed by mistake when the system is operating in sub-optimal state.
- If setting [RTC\\_CNTL\\_SWD\\_AUTO\\_FEED\\_EN](#) to 1, SWD controller can also feed SWD itself without any interaction with CPU or ULP-RISC-V.

After reset:

- Check [RTC\\_CNTL\\_RESET\\_CAUSE\\_PROCPU\[5:0\]](#) for the cause of CPU reset.  
If [RTC\\_CNTL\\_RESET\\_CAUSE\\_PROCPU\[5:0\]](#) == 0x12, it indicates that the cause is SWD reset.
- Set [RTC\\_CNTL\\_SWD\\_RST\\_FLAG\\_CLR](#) to clear the SWD reset flag.

## 13.5 Registers

MWDT registers are part of the timer submodule and are described in the Chapter 12: [64-bit Timers Timer Registers](#) section. RWDT and SWD registers are part of the RTC submodule and are described in Chapter 29: [Low-power Management RTC Registers](#) section.

## 14. XTAL32K Watchdog

### 14.1 Overview

XTAL32K Watchdog on ESP32-S2 is used to monitor the status of external crystal XTAL32K\_CLK. This Watchdog can detect the oscillation failure of XTAL32K\_CLK, change the clock source of RTC, etc. When XTAL32K\_CLK works as the clock source of RTC SLOW\_CLK and stops vibrating, XTAL32K Watchdog first switches to BACKUP32K\_CLK derived from RTC\_CLK and generates an interrupt (If the chip is in deep sleep mode, the CPU will be woken up), and then switches back to XTAL32K\_CLK after it is restarted by software.

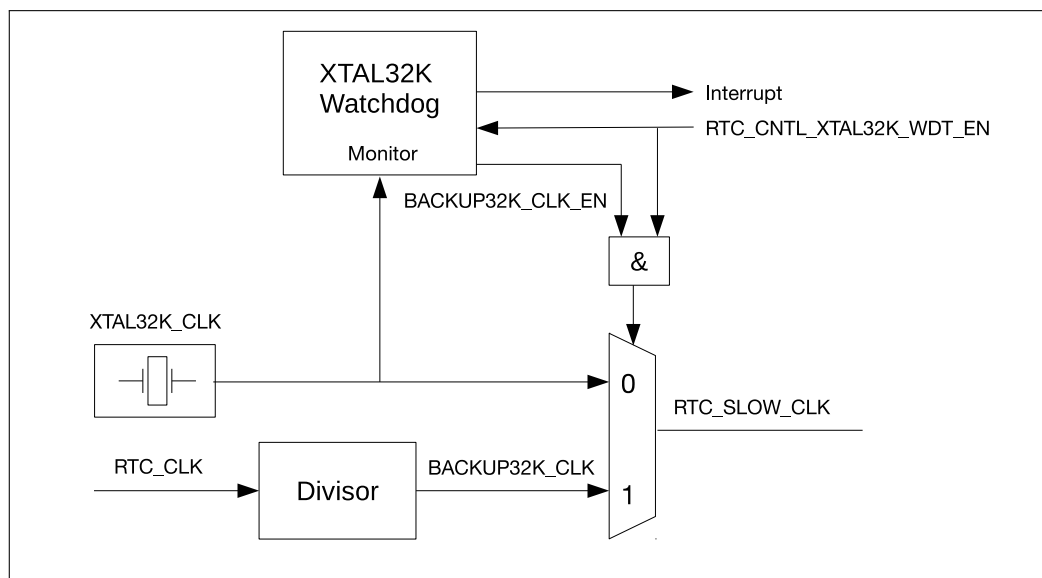


Figure 14-1. XTAL32K Watchdog

### 14.2 Features

#### 14.2.1 XTAL32K Watchdog Interrupts and Wake-up

When XTAL32K Watchdog detects the oscillation failure of XTAL32K\_CLK, an oscillation failure interrupt is generated. At this point the CPU will be woken up if in hibernation.

#### 14.2.2 BACKUP32K\_CLK

Once XTAL32K Watchdog detects the oscillation failure of XTAL32K\_CLK, it replaces XTAL32K\_CLK with BACKUP32K\_CLK (with a frequency of 32 kHz) derived from RTC\_CLK as RTC's SLOW\_CLK, so as to ensure proper functioning of the system.

### 14.3 Functional Description

#### 14.3.1 Workflow

1. XTAL32K Watchdog starts counting when `RTC_CNTL_XTAL32K_WDT_EN` is enabled. The counter keeps counting until it detects the positive edge of XTAL\_32K and is then cleared. When the counter reaches `RTC_CNTL_XTAL32K_WDT_TIMEOUT`, it generates an interrupt or a wake-up signal and is then reset.

2. XTAL32K Watchdog automatically enables BACKUP32K\_CLK as the alternative clock source of RTC SLOW\_CLK, to ensure the system's proper functioning and the accuracy of timers running on RTC SLOW\_CLK (e.g. RTC\_TIMER). For information about clock frequency configuration, please refer to Section [14.3.2 Configuring the Divisor of BACKUP32K\\_CLK](#).
3. To restore XTAL32K Watchdog, software restarts XTAL32K\_CLK by turning its XPD signal on and off via RTC\_CNTL\_XPD\_XTAL\_32K bit. Then, XTAL32K Watchdog switches back to XTAL32K\_CLK by setting [RTC\\_CNTL\\_XTAL32K\\_WDT\\_EN](#) bit to 0. If the chip is in deep sleep mode, XTAL32K Watchdog will wake up the CPU to finish the above steps.

### 14.3.2 Configuring the Divisor of BACKUP32K\_CLK

Chips have different RTC\_CLK frequencies due to production process variations. To ensure the accuracy of RTC\_TIMER and other timers running on SLOW\_CLK when BACKUP32K\_CLK is at work, the divisor of BACKUP32K\_CLK should be configured according to the actual frequency of RTC\_CLK.

Define the frequency of RTC\_CLK as  $f_{rtc\_clk}$  (unit: kHz), and the eight divisor components as  $x_0, x_1, x_2, x_3, x_4, x_5, x_6$ , and  $x_7$ , respectively.  $S = x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$ .

The following conditions should be fulfilled:

$$\begin{aligned} S &= f_{rtc\_clk} \times (4/32) \\ M + 1 &\geq x_n \geq M (0 \leq n \leq 7) \\ M &= f_{rtc\_clk}/32/2 \end{aligned}$$

$x_n$  should be an interger.  $M$  and  $S$  are rounded up or down. Each divisor component ( $x_0 \sim x_7$ ) is 4-bit long, and corresponds to the value of [RTC\\_CNTL\\_XTAL32K\\_CLK\\_FACTOR](#) (32-bit) in order.

For example, if the frequency of RTC\_CLK is 163 kHz, then  $f_{rtc\_clk} = 163$ ,  $S = 20$ ,  $M = 2$ , and  $\{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\} = \{2, 3, 2, 3, 2, 3, 2, 3\}$ . As a result, the frequency of BACKUP32K\_CLK is 32.6 kHz.

## 15. System Timer

### 15.1 Overview

System timer is a 64-bit timer specially for operating system. It can be used to schedule operating system tasks by generating periodical system ticks or certain time delay interrupts. With the help of RTC timer, system timer can keep updated after Light-sleep or Deep-sleep.

### 15.2 Main Features

- A 64-bit timer
- Clocked with APB\_CLK
- Timer value increment step can be configured for each APB\_CLK cycle.
- Support automatic time compensation in case of APB\_CLK clock source switching between PLL\_CLK and XTAL\_CLK, to improve timer accuracy.
- Generate three independent interrupts based on different alarm values or periods (targets).
- Support for 64-bit alarm values and 30-bit periods.
- Load back sleep time recorded by RTC timer after Deep-sleep or Light-sleep by software.
- Keep stalled if CPU is stalled or CPU is in on-chip-debugging mode.

### 15.3 Clock Source Selection

The System Timer is driven using XTAL\_CLK or PLL\_CLK clock. Selection between the two clock sources is described in Table CPU\_CLK Source in Chapter 2 *Reset and Clock*. For the specific clock frequency used for the System Timer, please refer to Table 11 *APB\_CLK Source*. On each clock period the timer will be increased by a step value configured in either `SYSTIMER_TIMER_XTAL_STEP` or `SYSTIMER_TIMER_PLL_STEP`, depending on which clock source is used.

### 15.4 Functional Description

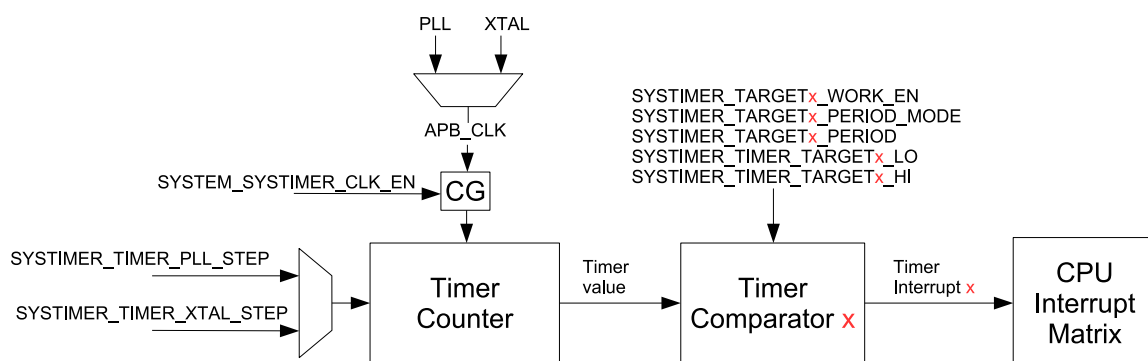


Figure 15-1. System Timer Structure

Figure 15-1 shows the structure of system timer. The system timer can be enabled by setting the bit `SYSTEM_SYSTIMER_CLK_EN` in register `SYSTEM_PERIP_CLK_EN0_REG` and be reset via software by setting



the bit `SYSTEM_SYSTIMER_RST` in register `SYSTEM_PERIP_RST_EN0_REG`. For more information, please refer to Table 33 *Peripheral Clock Gating and Reset Bits* in Chapter 6 *System Registers*.

#### 15.4.1 Read System Timer Value

1. Set `SYSTIMER_TIMER_UPDATE` to update the timer value into registers.
2. Wait till `SYSTIMER_TIMER_VALUE_VALID` is set, which means users now can read the timer values from registers.
3. Read the high 32 bits of the timer value from `SYSTIMER_TIMER_VALUE_HI`, and the low 32 bits from `SYSTIMER_TIMER_VALUE_LO`.

#### 15.4.2 Configure a Time-Delay Alarm

1. Read the current value of system timer, see Section 15.4.1. This value will be used to calculate the target in Step 3.
2. Clear `SYSTIMER_TARGETx_PERIOD_MODE` to set the timer into a time-delay alarm mode.
3. Write the high 32 bits of the target (alarm value) to `SYSTIMER_TIMER_TARGETx_HI`, and the low 32 bits to `SYSTIMER_TIMER_TARGETx_LO`.
4. Set `SYSTIMER_TARGETx_WORK_EN` to enable the selected work mode.
5. Set `SYSTIMER_INTx_ENA` to enable timer interrupt. When the timer counts to the alarm value, an interrupt will be triggered.

#### 15.4.3 Configure Periodic Alarms

1. Set `SYSTIMER_TARGETx_PERIOD_MODE` to configure the timer into periodic alarms mode.
2. Write the target (alarm period) to `SYSTIMER_TARGETx_PERIOD`.
3. Set `SYSTIMER_TARGETx_WORK_EN` to enable periodical alarms mode.
4. Set `SYSTIMER_INTx_ENA` to enable timer interrupt. An interrupt will be triggered when the timer counts to the target value set in Step 2.

#### 15.4.4 Update after Deep-sleep and Light-sleep

1. Configure RTC timer before the chip goes to Deep-sleep or Light-sleep, to record the exact sleep time.
2. Read the sleep time from RTC timer when the chip is woken up from Deep-sleep or Light-sleep.
3. Read current value of the system timer. See Section 15.4.1 for how to read the timer value.
4. Add current value of the system timer to the time that RTC timer records in Deep-sleep mode or in Light-sleep mode.
5. Write the result into `SYSTIMER_TIMER_LOAD_HI` (high 32 bits), and into `SYSTIMER_TIMER_LOAD_LO` (low 32 bits).
6. Set `SYSTIMER_TIMER_LOAD` to load the new value stored in `SYSTIMER_TIMER_LOAD_HI` and `SYSTIMER_TIMER_LOAD_LO` into the system timer. By such way, the system timer is updated.

## 15.5 Base Address

Users can access system timer registers with two base addresses, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter 1: [System and Memory](#).

**Table 55: System Timer Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F423000
PeriBUS2	0x60023000

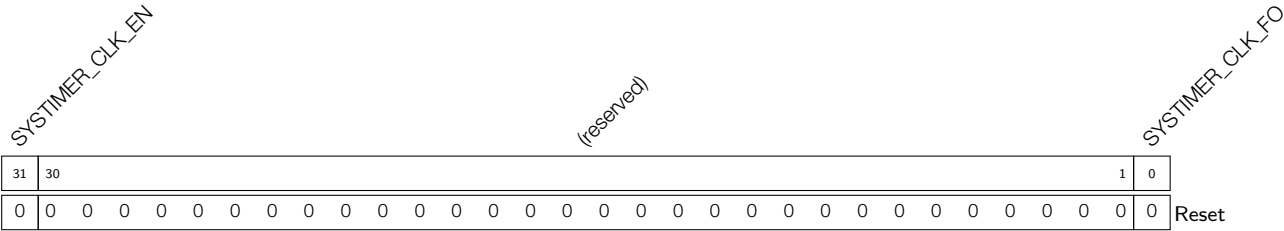
## 15.6 Register Summary

The addresses in the following table are relative to system timer base addresses provided in Section 15.5.

Name	Description	Address	Access
<b>System Timer Registers</b>			
<a href="#">SYSTIMER_CONF_REG</a>	Configure system timer clock	0x0000	R/W
<a href="#">SYSTIMER_LOAD_REG</a>	Load value to system timer	0x0004	WO
<a href="#">SYSTIMER_LOAD_HI_REG</a>	High 32 bits to be loaded to system timer	0x0008	R/W
<a href="#">SYSTIMER_LOAD_LO_REG</a>	Low 32 bits to be loaded to system timer	0x000C	R/W
<a href="#">SYSTIMER_STEP_REG</a>	System timer accumulation step	0x0010	R/W
<a href="#">SYSTIMER_TARGET0_HI_REG</a>	System timer target 0, high 32 bits	0x0014	R/W
<a href="#">SYSTIMER_TARGET0_LO_REG</a>	System timer target 0, low 32 bits	0x0018	R/W
<a href="#">SYSTIMER_TARGET1_HI_REG</a>	System timer target 1, high 32 bits	0x001C	R/W
<a href="#">SYSTIMER_TARGET1_LO_REG</a>	System timer target 1, low 32 bits	0x0020	R/W
<a href="#">SYSTIMER_TARGET2_HI_REG</a>	System timer target 2, high 32 bits	0x0024	R/W
<a href="#">SYSTIMER_TARGET2_LO_REG</a>	System timer target 2, low 32 bits	0x0028	R/W
<a href="#">SYSTIMER_TARGET0_CONF_REG</a>	Configure work mode for system timer target 0	0x002C	R/W
<a href="#">SYSTIMER_TARGET1_CONF_REG</a>	Configure work mode for system timer target 1	0x0030	R/W
<a href="#">SYSTIMER_TARGET2_CONF_REG</a>	Configure work mode for system timer target 2	0x0034	R/W
<a href="#">SYSTIMER_UPDATE_REG</a>	Read out system timer value	0x0038	varies
<a href="#">SYSTIMER_VALUE_HI_REG</a>	System timer value, high 32 bits	0x003C	RO
<a href="#">SYSTIMER_VALUE_LO_REG</a>	System timer value, low 32 bits	0x0040	RO
<a href="#">SYSTIMER_INT_ENA_REG</a>	System timer interrupt enable	0x0044	R/W
<a href="#">SYSTIMER_INT_RAW_REG</a>	System timer interrupt raw	0x0048	RO
<a href="#">SYSTIMER_INT_CLR_REG</a>	System timer interrupt clear	0x004C	WO
<b>Version Register</b>			
<a href="#">SYSTIMER_DATE_REG</a>	Version control register	0x00FC	R/W

# 15.7 Registers

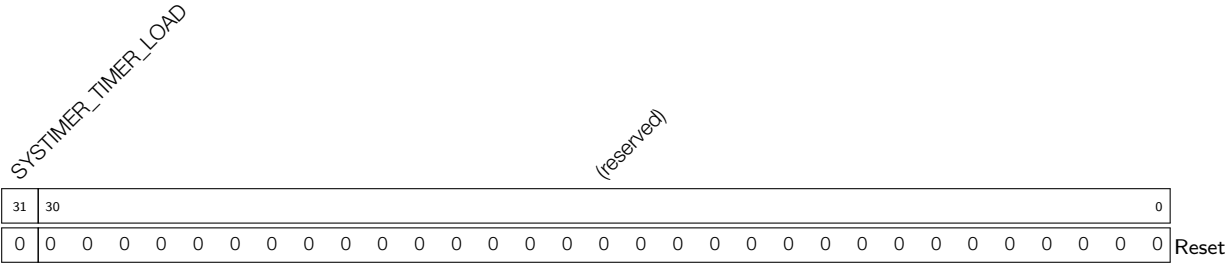
Register 15.1: SYSTIMER\_CONF\_REG (0x0000)



**SYSTIMER\_CLK\_FO** System timer clock force enable. (R/W)

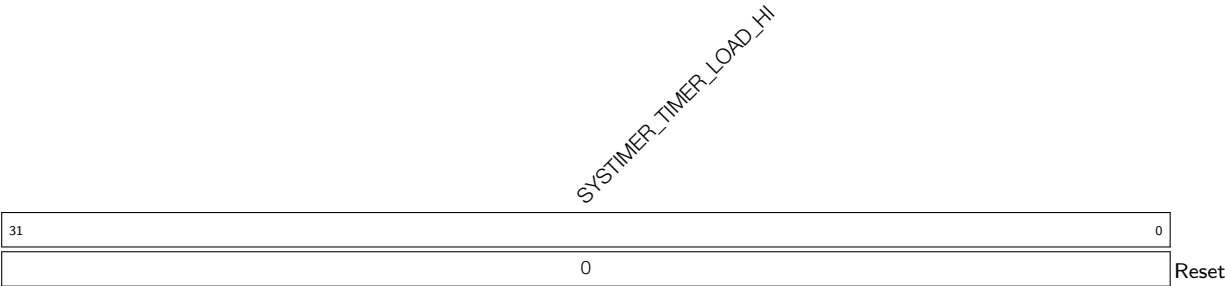
**SYSTIMER\_CLK\_EN** Register clock enable. (R/W)

Register 15.2: SYSTIMER\_LOAD\_REG (0x0004)



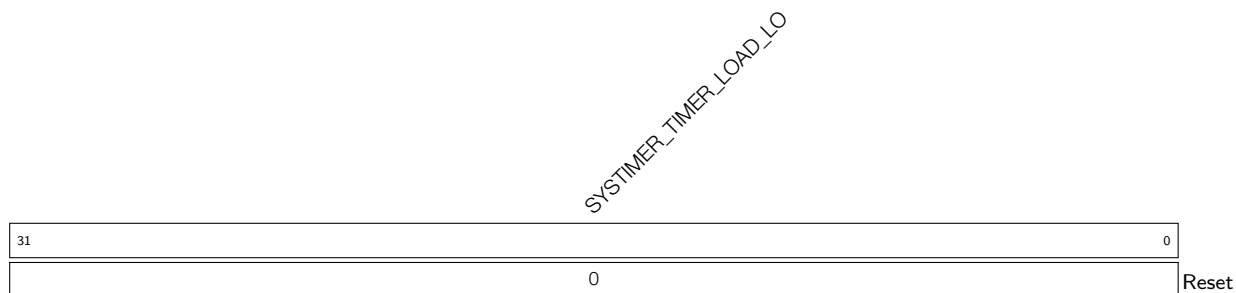
**SYSTIMER\_TIMER\_LOAD** Set this bit to 1, the value stored in [SYSTIMER\\_TIMER\\_LOAD\\_HI](#) and in [SYSTIMER\\_TIMER\\_LOAD\\_LO](#) will be loaded to system timer. (WO)

Register 15.3: SYSTIMER\_LOAD\_HI\_REG (0x0008)



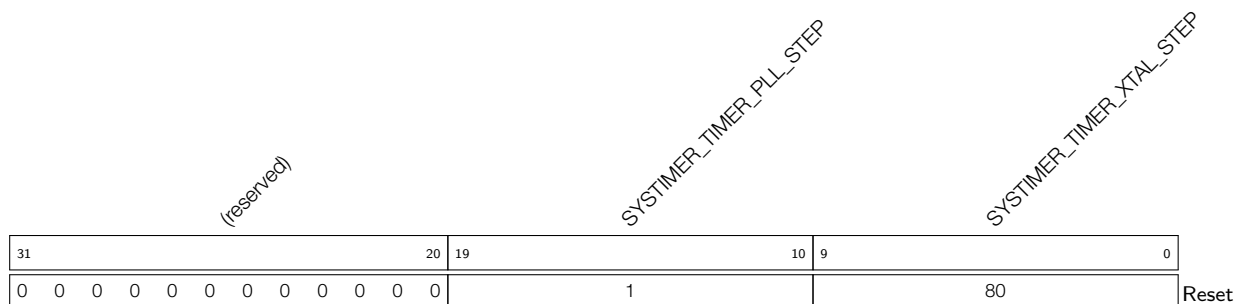
**SYSTIMER\_TIMER\_LOAD\_HI** The value to be loaded into system timer, high 32 bits. (R/W)

Register 15.4: SYSTIMER\_LOAD\_LO\_REG (0x000C)



**SYSTIMER\_TIMER\_LOAD\_LO** The value to be loaded into system timer, low 32 bits. (R/W)

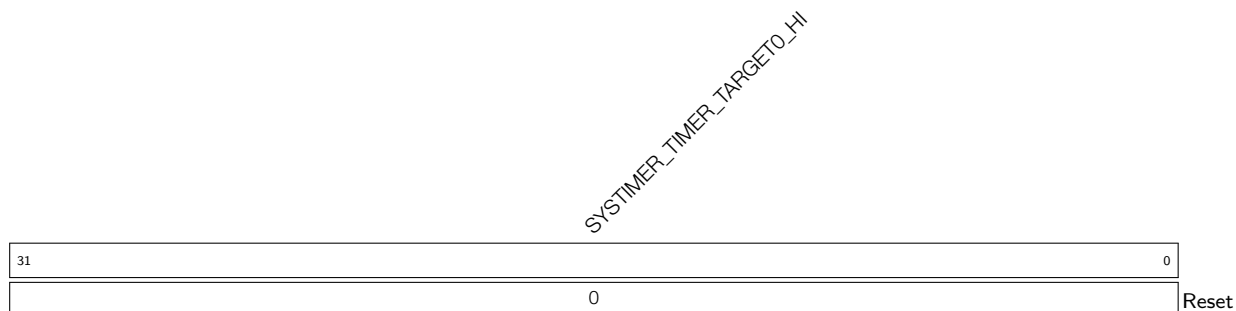
Register 15.5: SYSTIMER\_STEP\_REG (0x0010)



**SYSTIMER\_TIMER\_XTAL\_STEP** Set system timer increment step when using XTAL\_CLK. (R/W)

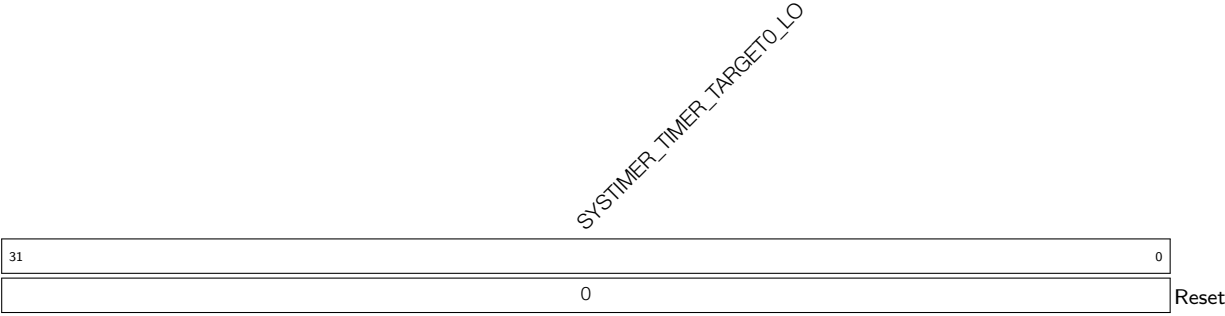
**SYSTIMER\_TIMER\_PLL\_STEP** Set system timer increment step when using PLL\_CLK. (R/W)

Register 15.6: SYSTIMER\_TARGET0\_HI\_REG (0x0014)



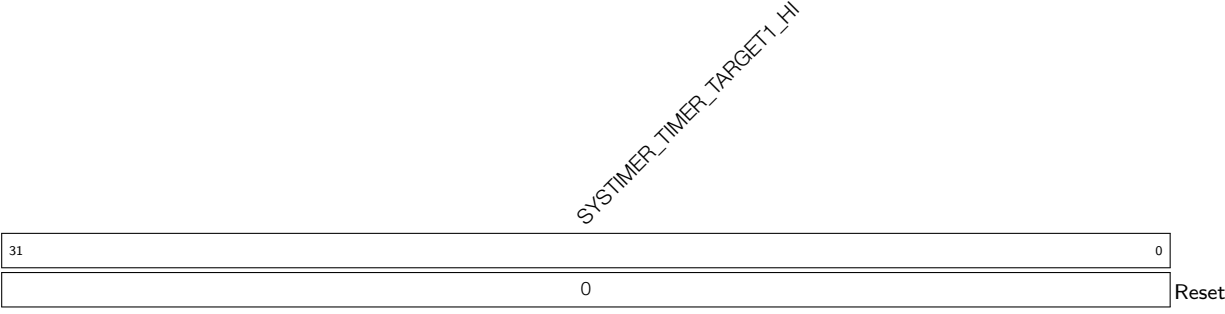
**SYSTIMER\_TIMER\_TARGET0\_HI** System timer target 0, high 32 bits. (R/W)

Register 15.7: SYSTIMER\_TARGET0\_LO\_REG (0x0018)



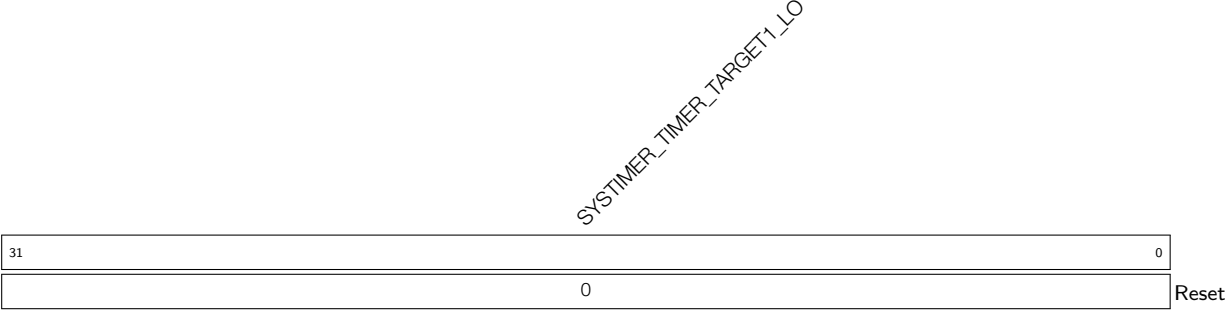
**SYSTIMER\_TIMER\_TARGET0\_LO** System timer target 0, low 32 bits. (R/W)

Register 15.8: SYSTIMER\_TARGET1\_HI\_REG (0x001C)



**SYSTIMER\_TIMER\_TARGET1\_HI** System timer target 1, high 32 bits. (R/W)

Register 15.9: SYSTIMER\_TARGET1\_LO\_REG (0x0020)



**SYSTIMER\_TIMER\_TARGET1\_LO** System timer target 1, low 32 bits. (R/W)

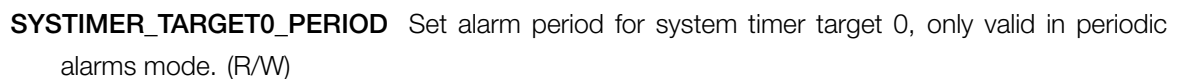
[Submit Documentation Feedback](#)



[Submit Documentation Feedback](#)



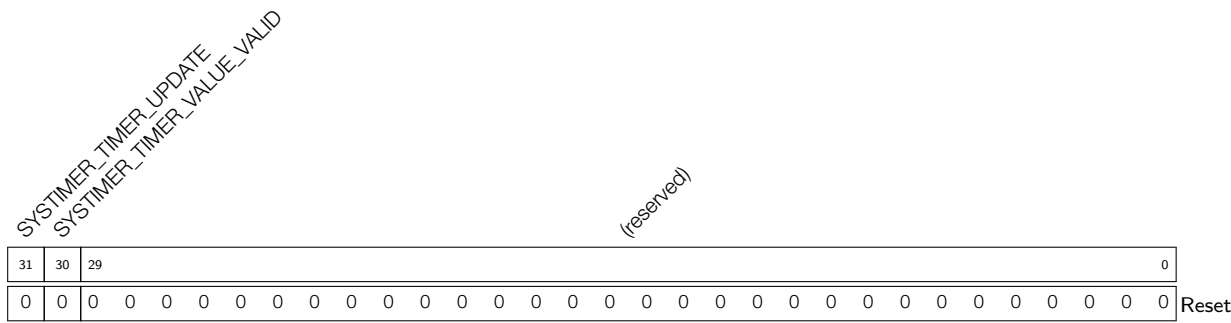
[Submit Documentation Feedback](#)



**SYSTIMER\_TARGET0\_WORK\_EN** System timer target 0 work enable. (R/W)



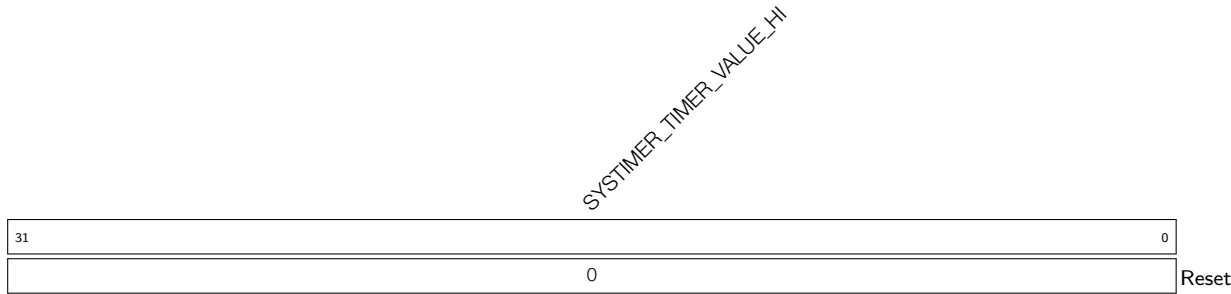
Register 15.15: SYSTIMER\_UPDATE\_REG (0x0038)



**SYSTIMER\_TIMER\_VALUE\_VALID** Check if it is valid to read out timer value from registers. 0: Not ready to read timer value from registers; 1: Ready to read timer value from registers. (RO)

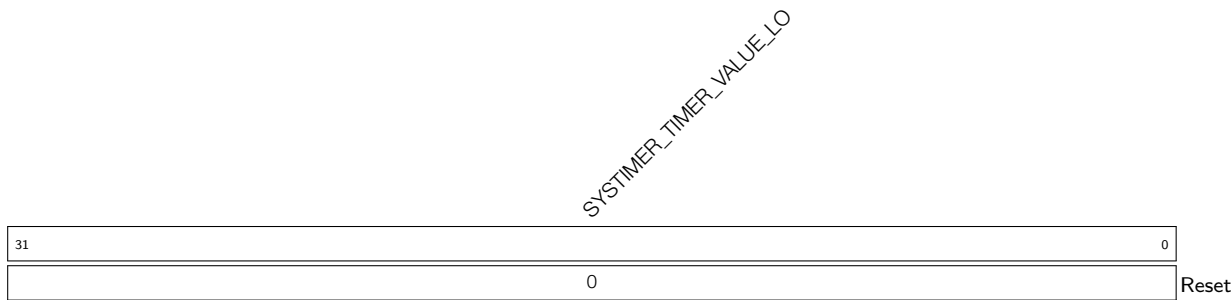
**SYSTIMER\_TIMER\_UPDATE** Update system timer value to registers. (WO)

Register 15.16: SYSTIMER\_VALUE\_HI\_REG (0x003C)



**SYSTIMER\_TIMER\_VALUE\_HI** System timer value, high 32 bits. (RO)

Register 15.17: SYSTIMER\_VALUE\_LO\_REG (0x0040)



**SYSTIMER\_TIMER\_VALUE\_LO** System timer value, low 32 bits. (RO)



**Register 15.18: SYSTIMER\_INT\_ENA\_REG (0x0044)**

(reserved)																															SYSTIMER_INT2_ENA SYSTIMER_INT1_ENA SYSTIMER_INT0_ENA																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
31																															3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**SYSTIMER\_INT0\_ENA** Interrupt enable bit of system timer target 0. (R/W)

**SYSTIMER\_INT1\_ENA** Interrupt enable bit of system timer target 1. (R/W)

**SYSTIMER\_INT2\_ENA** Interrupt enable bit of system timer target 2. (R/W)

**Register 15.19: SYSTIMER\_INT\_RAW\_REG (0x0048)**

(reserved)																												SYSTIMER_INT2_RAW SYSTIMER_INT1_RAW SYSTIMER_INT0_RAW																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
31																												3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**SYSTIMER\_INT0\_RAW** Interrupt raw bit of system timer target 0. (RO)

**SYSTIMER\_INT1\_RAW** Interrupt raw bit of system timer target 1. (RO)

**SYSTIMER\_INT2\_RAW** Interrupt raw bit of system timer target 2. (RO)

**Register 15.20: SYSTIMER\_INT\_CLR\_REG (0x004C)**

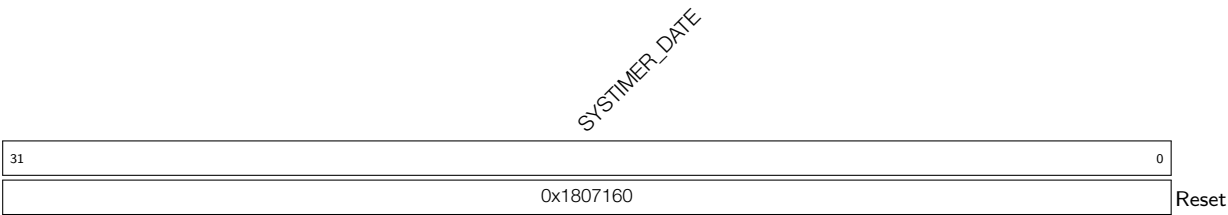
(reserved)																															SYSTIMER_INT2_CLR SYSTIMER_INT1_CLR SYSTIMER_INT0_CLR				
31																												3	2	1	0				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**SYSTIMER\_INT0\_CLR** Interrupt clear bit of system timer target 0. (WO)

**SYSTIMER\_INT1\_CLR** Interrupt clear bit of system timer target 1. (WO)

**SYSTIMER\_INT2\_CLR** Interrupt clear bit of system timer target 2. (WO)

Register 15.21: SYSTIMER\_DATE\_REG (0x00FC)



**SYSTIMER\_DATE** Version control register. (R/W)

## 16. eFuse Controller

### 16.1 Overview

ESP32-S2 has a 4096-bit eFuse that stores parameters in the SoC. Once an eFuse bit is programmed to 1, it can never be reverted to 0. Software can instruct the eFuse Controller to program individual bits for individual parameters as needed. Some of these parameters can be read by software using the eFuse Controller registers, while some can be directly used by hardware modules.

### 16.2 Features

- One-time programmable storage
- Configurable write protection
- Configurable software read protection
- Parameters use different hardware encoding schemes to protect against corruption

### 16.3 Functional Description

#### 16.3.1 Structure

There are 11 eFuse blocks (BLOCK0 ~ BLOCK10).

BLOCK0 holds most core system parameters. Among these parameters, 24 bits are invisible to software and can only be used by hardware and 38 bits are reserved for future use.

Table 57 lists all the parameters in BLOCK0 and their offsets, bit widths, functional description, as well as information on whether they can be used by hardware, and whether they are protected from programming.

The [EFUSE\\_WR\\_DIS](#) parameter is used to restrict the programming of other parameters, while [EFUSE\\_RD\\_DIS](#) is used to restrict software from reading BLOCK4 ~ BLOCK10. More information on these two parameters can be found in sections [16.3.1.1](#) and [16.3.1.2](#).

**Table 57: Parameters in BLOCK0**

Parameters	Offset	Bit Width	Hardware Use	Programming-Protection by <a href="#">EFUSE_WR_DIS</a> Bit Number	Description
<a href="#">EFUSE_WR_DIS</a>	0	32	Y	N/A	Disables programming of individual eFuses
<a href="#">EFUSE_RD_DIS</a>	32	7	Y	0	Disables software reading from individual eFuse blocks BLOCK4-10
<a href="#">EFUSE_DIS_RTC_RAM_BOOT</a>	39	1	N	1	Disables boot from RTC RAM
<a href="#">EFUSE_DIS_ICACHE</a>	40	1	Y	2	Disables ICache
<a href="#">EFUSE_DIS_DCACHE</a>	41	1	Y	2	Disables DCache
<a href="#">EFUSE_DIS_DOWNLOAD_ICACHE</a>	42	1	Y	2	Disables Icache when SoC is in Download mode
<a href="#">EFUSE_DIS_DOWNLOAD_DCACHE</a>	43	1	Y	2	Disables Dcache when SoC is in Download mode

Parameters	Offset	Bit Width	Hardware Use	Programming-Protection by EFUSE_WR_DIS Bit Number	Description
EFUSE_DIS_FORCE_DOWNLOAD	44	1	Y	2	Disables forcing chip into Download mode
EFUSE_DIS_USB	45	1	Y	2	Disables the USB OTG hardware
EFUSE_DIS_CAN	46	1	Y	2	Disables the TWAI Controller hardware
EFUSE_DIS_BOOT_REMAP	47	1	Y	2	Disables capability to Remap RAM to ROM address space
EFUSE_SOFT_DIS_JTAG	49	1	Y	2	Software disables JTAG. When software disabled, JTAG can be activated temporarily by HMAC peripheral.
EFUSE_HARD_DIS_JTAG	50	1	Y	2	Hardware disables JTAG permanently.
EFUSE_DIS_DOWNLOAD_MANUAL_ENCRYPT	51	1	Y	2	Disables flash encryption when in download boot modes
EFUSE_USB_EXCHG_PINS	56	1	Y	30	Exchanges USB D+ and D-pins
EFUSE_EXT_PHY_ENABLE	57	1	N	30	Enables external USB PHY
EFUSE_USB_FORCE_NOPERSIST	58	1	N	30	Forces to set USB BVALID to 1
EFUSE_VDD_SPI_XPD	68	1	Y	3	If VDD_SPI_FORCE is 1, determines if the VDD_SPI regulator is powered on
EFUSE_VDD_SPI_TIEH	69	1	Y	3	If VDD_SPI_FORCE is 1, determines VDD_SPI voltage. 0: VDD_SPI connects to 1.8 V LDO; 1: VDD_SPI connects to VDD_RTC_IO
EFUSE_VDD_SPI_FORCE	70	1	Y	3	When set, XPD_VDD_PSI_REG and VDD_SPI_TIEH will be used to configure VDD_SPI LDO
EFUSE_WDT_DELAY_SEL	80	2	Y	3	Selects RTC WDT timeout threshold at startup
EFUSE_SPI_BOOT_CRYPT_CNT	82	3	Y	4	Enables encryption and decryption, when an SPI boot mode is set. Feature is enabled when 1 or 3 bits are set in the eFuse, disabled otherwise.
EFUSE_SECURE_BOOT_KEY_REVOKE0	85	1	N	5	If set, revokes use of secure boot key digest 0

Parameters	Offset	Bit Width	Hardware Use	Programming-Protection by EFUSE_WR_DIS Bit Number	Description
EFUSE_SECURE_BOOT_KEY_REVOKE1	86	1	N	6	If set, revokes use of secure boot key digest 1
EFUSE_SECURE_BOOT_KEY_REVOKE2	87	1	N	7	If set, revokes use of secure boot key digest 2
EFUSE_KEY_PURPOSE_0	88	4	Y	8	KEY0 purpose, see Table 58
EFUSE_KEY_PURPOSE_1	92	4	Y	9	KEY1 purpose, see Table 58
EFUSE_KEY_PURPOSE_2	96	4	Y	10	KEY2 purpose, see Table 58
EFUSE_KEY_PURPOSE_3	100	4	Y	11	KEY3 purpose, see Table 58
EFUSE_KEY_PURPOSE_4	104	4	Y	12	KEY4 purpose, see Table 58
EFUSE_KEY_PURPOSE_5	108	4	Y	13	KEY5 purpose, see Table 58
EFUSE_SECURE_BOOT_EN	116	1	N	15	Enables secure boot
EFUSE_SECURE_BOOT_AGGRESSIVE_REVOKE	117	1	N	16	Enables aggressive secure boot key revocation mode
EFUSE_FLASH_TPUW	124	4	N	18	Configures flash startup delay after SoC power-up, unit is (ms/2). When the value is 15, delay is 7.5 ms.
EFUSE_DIS_DOWNLOAD_MODE	128	1	N	18	Disables all Download boot modes
EFUSE_DIS_LEGACY_SPI_BOOT	129	1	N	18	Disables Legacy SPI boot mode
EFUSE_UART_PRINT_CHANNEL	130	1	N	18	Selects the default UART for printing boot messages, 0: UART0; 1: UART1
EFUSE_DIS_USB_DOWNLOAD_MODE	132	1	N	18	Disables use of USB in UART download boot mode
EFUSE_ENABLE_SECURITY_DOWNLOAD	133	1	N	18	Enables secure UART download mode (read/write flash only)
EFUSE_UART_PRINT_CONTROL	134	2	N	18	Sets the default UART boot message output mode. 2'b00: Enabled; 2'b01: Enable when GPIO 46 is low at reset; 2'b10: Enable when GPIO 46 is high at rest; 2'b11: Disabled.
EFUSE_PIN_POWER_SELECTION	136	1	N	18	Sets default power supply for GPIO33 ~ GPIO37, set when SPI flash is initialized. 0: VDD3P3_CPU; 1: VDD_SPI
EFUSE_FLASH_TYPE	137	1	N	18	Selects SPI flash type, 0: maximum four data lines; 1: eight data lines

Parameters	Offset	Bit Width	Hardware Use	Programming-Protection by <a href="#">EFUSE_WR_DIS</a> Bit Number	Description
<a href="#">EFUSE_FORCE_SEND_RESUME</a>	138	1	N	18	Forces ROM code to send an SPI flash resume command during SPI boot
<a href="#">EFUSE_SECURE_VERSION</a>	139	16	N	18	Secure version (used by ESP-IDF anti-rollback feature)

Table 58 lists all key purpose and their values. Setting the eFuse parameter [EFUSE\\_KEY\\_PURPOSE\\_n](#) programs the purpose for eFuse block KEY $n$  ( $n$ : 0 ~ 5).

**Table 58: Key Purpose Values**

Key Purpose Values	Purposes
0	User purposes (software-only use)
1	Reserved
2	XTS_AES_256_KEY_1 (flash/PSRAM encryption)
3	XTS_AES_256_KEY_2 (flash/PSRAM encryption)
4	XTS_AES_128_KEY (flash/PSRAM encryption)
5	HMAC Downstream mode
6	JTAG soft enable key (uses HMAC Downstream mode)
7	Digital Signature peripheral key (uses HMAC Downstream mode)
8	HMAC Upstream mode
9	SECURE_BOOT_DIGEST0 (Secure Boot key digest)
10	SECURE_BOOT_DIGEST1 (Secure Boot key digest)
11	SECURE_BOOT_DIGEST2 (Secure Boot key digest)

Table 59 provides the details on the parameters in BLOCK1 ~ BLOCK10.

**Table 59: Parameters in BLOCK1-10**

BLOCK	Parameters	Bit Width	Hardware Use	Write Protection by <a href="#">EFUSE_WR_DIS</a> Bit Number	Software Read Protection by <a href="#">EFUSE_RD_DIS</a> Bit Number	Description
BLOCK1	<a href="#">EFUSE_MAC</a>	48	N	20	N/A	MAC address
	<a href="#">EFUSE_SPI_PAD_CONFIGURE</a>	[0:5]	N	20	N/A	CLK
		[6:11]	N	20	N/A	Q (D1)
		[12:17]	N	20	N/A	D (D0)
		[18:23]	N	20	N/A	CS
		[24:29]	N	20	N/A	HD (D3)
		[30:35]	N	20	N/A	WP (D2)
		[36:41]	N	20	N/A	DQS
		[42:47]	N	20	N/A	D4
		[48:53]	N	20	N/A	D5
		[54:59]	N	20	N/A	D6
		[60:65]	N	20	N/A	D7

BLOCK	Parameters	Bit Width	Hardware Use	Write Protection by <a href="#">EFUSE_WR_DIS</a> Bit Number	Software Read Protection by <a href="#">EFUSE_RD_DIS</a> Bit Number	Description
	<a href="#">EFUSE_SYS_DATA_PART0</a>	78	N	20	N/A	System data
BLOCK2	<a href="#">EFUSE_SYS_DATA_PART1</a>	256	N	21	N/A	System data
BLOCK3	<a href="#">EFUSE_USR_DATA</a>	256	N	22	N/A	User data
BLOCK4	<a href="#">EFUSE_KEY0_DATA</a>	256	Y	23	0	Key0 or user data
BLOCK5	<a href="#">EFUSE_KEY1_DATA</a>	256	Y	24	1	Key1 or user data
BLOCK6	<a href="#">EFUSE_KEY2_DATA</a>	256	Y	25	2	Key2 or user data
BLOCK7	<a href="#">EFUSE_KEY3_DATA</a>	256	Y	26	3	Key3 or user data
BLOCK8	<a href="#">EFUSE_KEY4_DATA</a>	256	Y	27	4	Key4 or user data
BLOCK9	<a href="#">EFUSE_KEY5_DATA</a>	256	Y	28	5	Key5 or user data
BLOCK10	<a href="#">EFUSE_SYS_DATA_PART2</a>	256	N	29	6	System data

Among these blocks, BLOCK4 ~ 9 stores KEY0 ~ 5, respectively. Up to six 256-bit keys can be programmed into eFuse. Whenever a key is programmed, its purpose value should also be programmed (see table 58). For example, a key for the JTAG function in HMAC Downstream mode is programmed to KEY3 (i.e., BLOCK7), then, the key purpose value 6 should be programmed to [EFUSE\\_KEY\\_PURPOSE\\_3](#).

BLOCK1 ~ BLOCK10 use the RS coding scheme, so there are some restrictions on writing to these parameters (refer to Section 16.3.1.3: *Data Storage* and Section 16.3.2: *Software Programming of Parameters*).

### 16.3.1.1 EFUSE\_WR\_DIS

Parameter [EFUSE\\_WR\\_DIS](#) determines whether individual eFuse parameters are write-protected. After burning [EFUSE\\_WR\\_DIS](#), execute an eFuse read operation so the new values will take effect (refer to *Updating eFuse read registers* in Section 16.3.3).

The columns “Programming-Protected by [EFUSE\\_WR\\_DIS](#)” in Table 57 and Table 59 list the specific bits of [EFUSE\\_WR\\_DIS](#) that determine the write protected status of each parameter.

When the corresponding bit is 0, the parameter is not write protected and can be programmed if the parameter has not been programmed.

When the corresponding bit is 1, the parameter is write protected and none of its bits can be modified. The non-programmed bits always remain 0, while programmed bits always remain 1.

### 16.3.1.2 EFUSE\_RD\_DIS

Only the eFuse blocks BLOCK4 ~ BLOCK10 can be individually software read protected. The corresponding bit in [EFUSE\\_RD\\_DIS](#) is shown in Table 59. After burning [EFUSE\\_RD\\_DIS](#), execute an eFuse read operation so the new values will take effect (refer to *Updating eFuse read registers* in Section 16.3.3).

If the corresponding `EFUSE_RD_DIS` bit is 0, then the eFuse block can be read by software. If the corresponding `EFUSE_RD_DIS` bit is 1, then the parameter controlled by this bit is software read protected.

Other parameters that are not in BLOCK4 ~ BLOCK10 can always be read by software.

Although BLOCK4 ~ BLOCK10 can be set to read-protected, they can still be used by hardware modules, if the `EFUSE_KEY_PURPOSE_n` bit is set accordingly.

### 16.3.1.3 Data Storage

Internal to the SoC, eFuses use hardware encoding schemes to protect against data corruption.

All BLOCK0 parameters except for `EFUSE_WR_DIS` are stored with four backups, meaning each bit is stored four times. This backup scheme is automatically applied by the hardware and is not visible to software.

BLOCK1 ~ BLOCK10 use RS (44, 32) coding scheme that supports up to 5 bytes of automatic error correction. The primitive polynomial of RS (44, 32) is  $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ . The shift register circuit that generates the check code is shown in Figure 16-1, where `gf_mul_n` (n is an integer) is the result of multiplying a byte of data in the  $GF(2^8)$  field by the element  $\alpha^n$ .

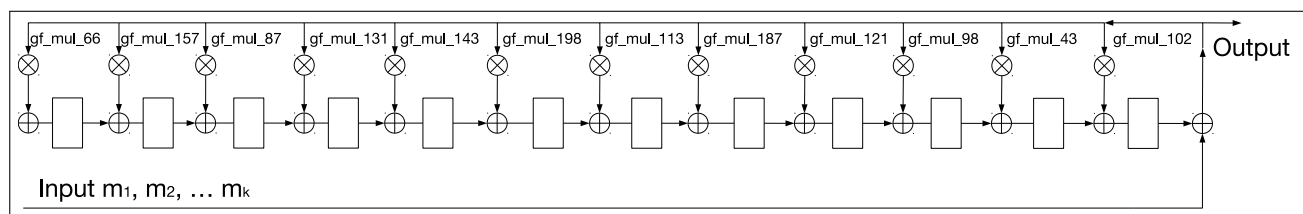


Figure 16-1. Shift Register Circuit

Software must encode the 32-byte parameter using RS (44, 32) to generate a 12-byte check code, and then burn the parameter and the check code into eFuse at the same time. The eFuse Controller automatically decodes the RS encoding and applies error correction when reading back the eFuse block.

Because the RS check codes are generated across the entire 256-bit eFuse block, each block can only be written to one time.

### 16.3.2 Software Programming of Parameters

The eFuse controller can only write to eFuse parameters in one block at a time. BLOCK0 ~ BLOCK10 share the same registers to store the parameters to be programmed. Configure parameter `EFUSE_BLK_NUM` to indicate which block is to be programmed.

#### Programming BLOCK0

When `EFUSE_BLK_NUM` = 0, BLOCK0 is programmed. Register `EFUSE_PGM_DATA0_REG` stores `EFUSE_WR_DIS`. `EFUSE_PGM_DATA1_REG` ~ `EFUSE_PGM_DATA5_REG` store the information of new BLOCK0 parameters to be programmed. Note that 24 BLOCK0 bits are reserved and can only be used by hardware. These must always be set to 0 in the programming registers. The specific bits are:

- `EFUSE_PGM_DATA1_REG`[29:31]
- `EFUSE_PGM_DATA1_REG`[20:23]
- `EFUSE_PGM_DATA2_REG`[7:15]
- `EFUSE_PGM_DATA2_REG`[0:3]



- [EFUSE\\_PGM\\_DATA3\\_REG](#)[16:19]

Values written to [EFUSE\\_PGM\\_DATA6\\_REG](#) ~ [EFUSE\\_PGM\\_DATA7\\_REG](#) and [EFUSE\\_PGM\\_CHECK\\_VALUE0\\_REG](#) ~ [EFUSE\\_PGM\\_CHECK\\_VALUE2\\_REG](#) are ignored when programming BLOCK0.

### Programming BLOCK1

When [EFUSE\\_BLK\\_NUM](#) = 1, [EFUSE\\_PGM\\_DATA0\\_REG](#) ~ [EFUSE\\_PGM\\_DATA5\\_REG](#) store the parameters to be programmed. [EFUSE\\_PGM\\_CHECK\\_VALUE0\\_REG](#) ~ [EFUSE\\_PGM\\_DATA2\\_REG](#) store the corresponding RS check codes. Values written to [EFUSE\\_PGM\\_DATA6\\_REG](#) ~ [EFUSE\\_PGM\\_DATA7\\_REG](#) are ignored when programming BLOCK1, the RS check codes should be calculated as if these bits were all 0.

### Programming BLOCK2 ~ 10

When [EFUSE\\_BLK\\_NUM](#) = 2 ~ 10, [EFUSE\\_PGM\\_DATA0\\_REG](#) ~ [EFUSE\\_PGM\\_DATA7\\_REG](#) store the parameters to be programmed to this block. [EFUSE\\_PGM\\_CHECK\\_VALUE0\\_REG](#) ~ [EFUSE\\_PGM\\_CHECK\\_VALUE2\\_REG](#) store the corresponding RS check codes.

### Programming process

The process of programming parameters is as follows:

1. Set [EFUSE\\_BLK\\_NUM](#) parameter as described above.
2. Write the parameters to be programmed into registers [EFUSE\\_PGM\\_DATA0\\_REG](#) ~ [EFUSE\\_PGM\\_DATA7\\_REG](#) and [EFUSE\\_PGM\\_CHECK\\_VALUE0\\_REG](#) ~ [EFUSE\\_PGM\\_CHECK\\_VALUE2\\_REG](#).
3. Ensure the eFuse clock registers are set correctly as described in Section [16.3.4.1: eFuse-Programming Timing](#).
4. Ensure the eFuse programming voltage VDDQ is set correctly as described in Section [16.3.4.2: eFuse VDDQ Setting](#).
5. Set [EFUSE\\_OP\\_CODE](#) field in register [EFUSE\\_CONF\\_REG](#) to 0x5A5A.
6. Set [EFUSE\\_PGM\\_CMD](#) field in register [EFUSE\\_CMD\\_REG](#) to 1.
7. Poll register [EFUSE\\_CMD\\_REG](#) until it is 0x0, or wait for a `pgm_done` interrupt. Information on how to identify a `pgm/read_done` interrupt is provided at the end of Section [16.3.3](#).
8. Clear the parameters written into the register.
9. Trigger an eFuse read operation (see Section [16.3.3: Software Reading of Parameters](#)) to update eFuse registers with the new values.

### Limitations

For BLOCK0, the programming of different parameters and even the programming of different bits of the same parameter does not need to be done at once. It is, however, recommended that users minimize programming cycles and program all the bits of a parameter in one programming action. In addition, after all parameters controlled by a certain bit of [EFUSE\\_WR\\_DIS](#) are programmed, that bit should be immediately programmed. The programming of parameters controlled by a certain bit of [EFUSE\\_WR\\_DIS](#), and the programming of the bit itself can even be completed at the same time. Repeated programming of already programmed bits is strictly forbidden, otherwise, programming errors will occur.

BLOCK1 cannot be programmed by users as it has been programmed at manufacturing.

BLOCK2 ~ 10 can only be programmed once. Repeated programming is not allowed.

### 16.3.3 Software Reading of Parameters

Software cannot read eFuse bits directly. The eFuse Controller hardware reads all eFuse bits and stores the results to their corresponding registers in the memory space. Then, software can read eFuse bits by reading the registers that start with EFUSE\_RD\_. Details are provided in the table below.

BLOCK	Read Registers	When Programming This Block
0	<a href="#">EFUSE_RD_WR_DIS_REG</a>	<a href="#">EFUSE_PGM_DATA0_REG</a>
0	EFUSE_RD_REPEAT_DATA0 ~ 4_REG	EFUSE_PGM_DATA1 ~ 5_REG
1	EFUSE_RD_MAC_SPI_SYS_0 ~ 5_REG	EFUSE_PGM_DATA0 ~ 5_REG
2	EFUSE_RD_SYS_DATA_PART1_0 ~ 7_REG	EFUSE_PGM_DATA0 ~ 7_REG
3	EFUSE_RD_USR_DATA0 ~ 7_REG	EFUSE_PGM_DATA0 ~ 7_REG
4-9	EFUSE_RD_KEY <sub><i>n</i></sub> _DATA0 ~ 7_REG ( <i>n</i> : 0 ~ 5)	EFUSE_PGM_DATA0 ~ 7_REG
10	EFUSE_RD_SYS_DATA_PART2_0 ~ 7_REG	EFUSE_PGM_DATA0 ~ 7_REG

#### Updating eFuse read registers

The eFuse Controller hardware populates the read registers from the internal eFuse storage. This read operation happens on system reset and can also be triggered manually by software as needed, for example if new eFuse values have been programmed.

The process of triggering an eFuse controller read from software is as follows:

1. Configure the eFuse read timing registers as described in Section [16.3.4.3: eFuse-Read Timing](#).
2. Set the [EFUSE\\_OP\\_CODE](#) field in register [EFUSE\\_CONF\\_REG](#) to 0x5AA5.
3. Set the [EFUSE\\_READ\\_CMD](#) field in register [EFUSE\\_CMD\\_REG](#) to 1.
4. Poll the [EFUSE\\_CMD\\_REG](#) register until it is 0x0, or wait for a read\_done interrupt. Information on how to identify a pgm/read\_done interrupt is provided below.
5. Software reads the values of each parameter from memory.

The eFuse read registers will now hold updated values for all eFuse parameters.

#### Error detection

Error registers allow software to detect an inconsistency in the stored eFuses.

EFUSE\_RD\_REPEAT\_ERR0 ~ 3\_REG indicate inconsistencies in the stored backup copies of the parameters in BLOCK0 (except for [EFUSE\\_WR\\_DIS](#)). Value 1 indicates an error was detected, and the bit became invalid. Value 0 indicates no error.

Registers EFUSE\_RD\_RS\_ERR0 ~ 1\_REG store the number of corrected bytes as well as the result of RS decoding during eFuse reading BLOCK1 ~ BLOCK10.

Software can read the values of above registers only after the eFuse read registers have been updated.

#### Identifying program/read operation completion

The two methods to identify the completion of program/read operation are described below. Please note that bit 1 corresponds to program operation, and bit 0 corresponds to read operation.

- Method one:

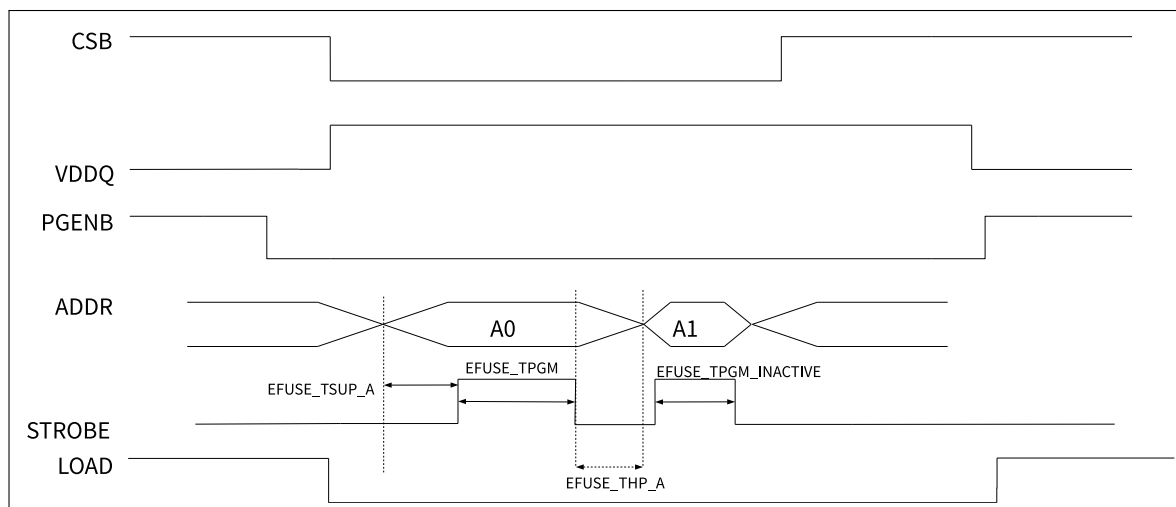
1. Poll bit 1/0 in register [EFUSE\\_INT\\_RAW\\_REG](#) until it is 1, which represents the completion of a program/read operation.
- Method two:
    1. Set bit 1/0 in register [EFUSE\\_INT\\_ENA\\_REG](#) to 1 to enable eFuse Controller to post a pgm/read\_done interrupt.
    2. Configure Interrupt Matrix to enable the CPU to respond to eFuse interrupt signal.
    3. Wait for the pgm/read\_done interrupt.
    4. Set the bit 1/0 in register [EFUSE\\_INT\\_CLR\\_REG](#) to 1 to clear the pgm/read\_done interrupt.

### 16.3.4 Timing

#### 16.3.4.1 eFuse-Programming Timing

Figure 16-2 shows the timing for programming eFuse. Four registers [EFUSE\\_TSUP\\_A](#), [EFUSE\\_TPGM](#), [EFUSE\\_THP\\_A](#), and [EFUSE\\_TPGM\\_INACTIVE](#) are used to configure the timing. Terms used in the timing diagrams in this section are described as follows:

- CSB: Chip select, active low
- VDDQ: eFuse programming voltage
- PGENB: eFuse programming enable signal, active low



**Figure 16-2. eFuse-Programming Timing Diagram**

The eFuse block uses the CLK\_APB clock, which is configurable. Therefore, the timing parameters should be configured according to the specific clock frequency. After reset, the initial parameters are based on 20 MHz clock frequency.

**Table 61: Configuration of eFuse-Programming Timing Parameters**

APB Frequency	<a href="#">EFUSE_TSUP_A</a> (> 6.669 ns)	<a href="#">EFUSE_TPGM</a> (9-11 $\mu$ s, usually 10 $\mu$ s)	<a href="#">EFUSE_THP_A</a> (> 6.166 ns)	<a href="#">EFUSE_TPGM_INACTIVE</a> (> 35.96 ns)
80 MHz	0x2	0x320	0x2	0x4
40 MHz	0x1	0x190	0x1	0x2
20 MHz	0x1	0xC8	0x1	0x1

In Figure 16-2, Address A0 is programmed, then the corresponding eFuse bit is 1; Address A1 is not programmed, then the corresponding eFuse bit is 0.

#### 16.3.4.2 eFuse VDDQ Timing Setting

VDDQ is the eFuse programming voltage, and its timing parameters should be configured according to the APB clock frequency:

**Table 62: Configuration of VDDQ Timing Parameters**

APB Frequency	EFUSE_DAC_CLK_DIV ( $> 1 \mu\text{s}$ )	EFUSE_PWR_ON_NUM ( $> \text{EFUSE\_DAC\_CLK\_DIV} \times 255$ )	EFUSE_PWR_OFF_NUM ( $> 3 \mu\text{s}$ )
80 MHz	0xA0	0xA200	0x100
40 MHz	0x50	0x5100	0x80
20 MHz	0x28	0x2880	0x40

#### 16.3.4.3 eFuse-Read Timing

Figure 16-3 shows the timing for reading eFuse. Three registers EFUSE\_TSR\_A, EFUSE\_TRD, and EFUSE\_THR\_A are used to configure the timing.

The parameters should be configured according to the specific APB clock frequency. Details can be found in the table below.

**Table 63: Configuration of eFuse-Reading Parameters**

APB Frequency	EFUSE_TSR_A ( $> 6.669 \text{ ns}$ )	EFUSE_TRD ( $> 35.96 \text{ ns}$ )	EFUSE_THR_A ( $> 6.166 \text{ ns}$ )
80 MHz	0x2	0x4	0x2
40 MHz	0x1	0x2	0x1
20 MHz	0x1	0x1	0x1

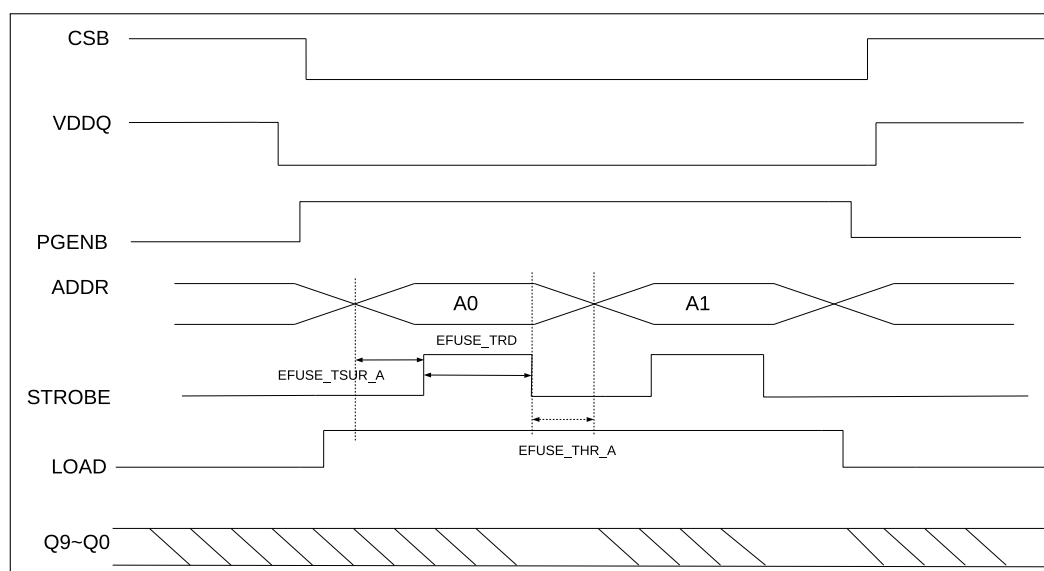


Figure 16-3. Timing Diagram for Reading eFuse

### 16.3.5 The Use of Parameters by Hardware Modules

Hardware modules are directly hardwired to the ESP32-S2 in order to use the parameters listed in Table 57 and 59, specifically those marked with “Y” in columns “Hardware Use”. Software cannot change this behavior.

### 16.3.6 Interrupts

- `pgm_done` interrupt: Triggered when eFuse programming has finished. To enable this interrupt, set `EFUSE_PGM_DONE_INT_ENA` to 1.
- `read_done` interrupt: Triggered when eFuse reading has finished. To enable this interrupt, set `EFUSE_READ_DONE_INT_ENA` to 1.

## 16.4 Base Address

Users can access the eFuse Controller with two base addresses, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter 1: *System and Memory*.

Table 64: eFuse Controller Base Address

Bus to Access Peripheral	Base Address
PeriBUS1	0x6001A000
PeriBUS2	0x3FC1A000

## 16.5 Register Summary

The addresses in the following table are relative to the eFuse base addresses provided in Section 16.4.

Name	Description	Address	Access
<b>PGM Data Registers</b>			
<a href="#">EFUSE_PGM_DATA0_REG</a>	Register 0 that stores data to be programmed.	0x0000	R/W
<a href="#">EFUSE_PGM_DATA1_REG</a>	Register 1 that stores data to be programmed.	0x0004	R/W

Name	Description	Address	Access
EFUSE_PGM_DATA2_REG	Register 2 that stores data to be programmed.	0x0008	R/W
EFUSE_PGM_DATA3_REG	Register 3 that stores data to be programmed.	0x000C	R/W
EFUSE_PGM_DATA4_REG	Register 4 that stores data to be programmed.	0x0010	R/W
EFUSE_PGM_DATA5_REG	Register 5 that stores data to be programmed.	0x0014	R/W
EFUSE_PGM_DATA6_REG	Register 6 that stores data to be programmed.	0x0018	R/W
EFUSE_PGM_DATA7_REG	Register 7 that stores data to be programmed.	0x001C	R/W
EFUSE_PGM_CHECK_VALUE0_REG	Register 0 that stores the RS code to be programmed.	0x0020	R/W
EFUSE_PGM_CHECK_VALUE1_REG	Register 1 that stores the RS code to be programmed.	0x0024	R/W
EFUSE_PGM_CHECK_VALUE2_REG	Register 2 that stores the RS code to be programmed.	0x0028	R/W
<b>Read Data Registers</b>			
EFUSE_RD_WR_DIS_REG	Register 0 of BLOCK0.	0x002C	RO
EFUSE_RD_REPEAT_DATA0_REG	Register 1 of BLOCK0.	0x0030	RO
EFUSE_RD_REPEAT_DATA1_REG	Register 2 of BLOCK0.	0x0034	RO
EFUSE_RD_REPEAT_DATA2_REG	Register 3 of BLOCK0.	0x0038	RO
EFUSE_RD_REPEAT_DATA3_REG	Register 4 of BLOCK0.	0x003C	RO
EFUSE_RD_REPEAT_DATA4_REG	Register 5 of BLOCK0.	0x0040	RO
EFUSE_RD_MAC_SPI_SYS_0_REG	Register 0 of BLOCK1.	0x0044	RO
EFUSE_RD_MAC_SPI_SYS_1_REG	Register 1 of BLOCK1.	0x0048	RO
EFUSE_RD_MAC_SPI_SYS_2_REG	Register 2 of BLOCK1.	0x004C	RO
EFUSE_RD_MAC_SPI_SYS_3_REG	Register 3 of BLOCK1.	0x0050	RO
EFUSE_RD_MAC_SPI_SYS_4_REG	Register 4 of BLOCK1.	0x0054	RO
EFUSE_RD_MAC_SPI_SYS_5_REG	Register 5 of BLOCK1.	0x0058	RO
EFUSE_RD_SYS_DATA_PART1_0_REG	Register 0 of BLOCK2 (system).	0x005C	RO
EFUSE_RD_SYS_DATA_PART1_1_REG	Register 1 of BLOCK2 (system).	0x0060	RO
EFUSE_RD_SYS_DATA_PART1_2_REG	Register 2 of BLOCK2 (system).	0x0064	RO
EFUSE_RD_SYS_DATA_PART1_3_REG	Register 3 of BLOCK2 (system).	0x0068	RO
EFUSE_RD_SYS_DATA_PART1_4_REG	Register 4 of BLOCK2 (system).	0x006C	RO
EFUSE_RD_SYS_DATA_PART1_5_REG	Register 5 of BLOCK2 (system).	0x0070	RO
EFUSE_RD_SYS_DATA_PART1_6_REG	Register 6 of BLOCK2 (system).	0x0074	RO
EFUSE_RD_SYS_DATA_PART1_7_REG	Register 7 of BLOCK2 (system).	0x0078	RO
EFUSE_RD_USR_DATA0_REG	Register 0 of BLOCK3 (user).	0x007C	RO
EFUSE_RD_USR_DATA1_REG	Register 1 of BLOCK3 (user).	0x0080	RO
EFUSE_RD_USR_DATA2_REG	Register 2 of BLOCK3 (user).	0x0084	RO
EFUSE_RD_USR_DATA3_REG	Register 3 of BLOCK3 (user).	0x0088	RO
EFUSE_RD_USR_DATA4_REG	Register 4 of BLOCK3 (user).	0x008C	RO
EFUSE_RD_USR_DATA5_REG	Register 5 of BLOCK3 (user).	0x0090	RO
EFUSE_RD_USR_DATA6_REG	Register 6 of BLOCK3 (user).	0x0094	RO
EFUSE_RD_USR_DATA7_REG	Register 7 of BLOCK3 (user).	0x0098	RO
EFUSE_RD_KEY0_DATA0_REG	Register 0 of BLOCK4 (KEY0).	0x009C	RO
EFUSE_RD_KEY0_DATA1_REG	Register 1 of BLOCK4 (KEY0).	0x00A0	RO

Name	Description	Address	Access
<a href="#">EFUSE_RD_KEY0_DATA2_REG</a>	Register 2 of BLOCK4 (KEY0).	0x00A4	RO
<a href="#">EFUSE_RD_KEY0_DATA3_REG</a>	Register 3 of BLOCK4 (KEY0).	0x00A8	RO
<a href="#">EFUSE_RD_KEY0_DATA4_REG</a>	Register 4 of BLOCK4 (KEY0).	0x00AC	RO
<a href="#">EFUSE_RD_KEY0_DATA5_REG</a>	Register 5 of BLOCK4 (KEY0).	0x00B0	RO
<a href="#">EFUSE_RD_KEY0_DATA6_REG</a>	Register 6 of BLOCK4 (KEY0).	0x00B4	RO
<a href="#">EFUSE_RD_KEY0_DATA7_REG</a>	Register 7 of BLOCK4 (KEY0).	0x00B8	RO
<a href="#">EFUSE_RD_KEY1_DATA0_REG</a>	Register 0 of BLOCK5 (KEY1).	0x00BC	RO
<a href="#">EFUSE_RD_KEY1_DATA1_REG</a>	Register 1 of BLOCK5 (KEY1).	0x00C0	RO
<a href="#">EFUSE_RD_KEY1_DATA2_REG</a>	Register 2 of BLOCK5 (KEY1).	0x00C4	RO
<a href="#">EFUSE_RD_KEY1_DATA3_REG</a>	Register 3 of BLOCK5 (KEY1).	0x00C8	RO
<a href="#">EFUSE_RD_KEY1_DATA4_REG</a>	Register 4 of BLOCK5 (KEY1).	0x00CC	RO
<a href="#">EFUSE_RD_KEY1_DATA5_REG</a>	Register 5 of BLOCK5 (KEY1).	0x00D0	RO
<a href="#">EFUSE_RD_KEY1_DATA6_REG</a>	Register 6 of BLOCK5 (KEY1).	0x00D4	RO
<a href="#">EFUSE_RD_KEY1_DATA7_REG</a>	Register 7 of BLOCK5 (KEY1).	0x00D8	RO
<a href="#">EFUSE_RD_KEY2_DATA0_REG</a>	Register 0 of BLOCK6 (KEY2).	0x00DC	RO
<a href="#">EFUSE_RD_KEY2_DATA1_REG</a>	Register 1 of BLOCK6 (KEY2).	0x00E0	RO
<a href="#">EFUSE_RD_KEY2_DATA2_REG</a>	Register 2 of BLOCK6 (KEY2).	0x00E4	RO
<a href="#">EFUSE_RD_KEY2_DATA3_REG</a>	Register 3 of BLOCK6 (KEY2).	0x00E8	RO
<a href="#">EFUSE_RD_KEY2_DATA4_REG</a>	Register 4 of BLOCK6 (KEY2).	0x00EC	RO
<a href="#">EFUSE_RD_KEY2_DATA5_REG</a>	Register 5 of BLOCK6 (KEY2).	0x00F0	RO
<a href="#">EFUSE_RD_KEY2_DATA6_REG</a>	Register 6 of BLOCK6 (KEY2).	0x00F4	RO
<a href="#">EFUSE_RD_KEY2_DATA7_REG</a>	Register 7 of BLOCK6 (KEY2).	0x00F8	RO
<a href="#">EFUSE_RD_KEY3_DATA0_REG</a>	Register 0 of BLOCK7 (KEY3).	0x00FC	RO
<a href="#">EFUSE_RD_KEY3_DATA1_REG</a>	Register 1 of BLOCK7 (KEY3).	0x0100	RO
<a href="#">EFUSE_RD_KEY3_DATA2_REG</a>	Register 2 of BLOCK7 (KEY3).	0x0104	RO
<a href="#">EFUSE_RD_KEY3_DATA3_REG</a>	Register 3 of BLOCK7 (KEY3).	0x0108	RO
<a href="#">EFUSE_RD_KEY3_DATA4_REG</a>	Register 4 of BLOCK7 (KEY3).	0x010C	RO
<a href="#">EFUSE_RD_KEY3_DATA5_REG</a>	Register 5 of BLOCK7 (KEY3).	0x0110	RO
<a href="#">EFUSE_RD_KEY3_DATA6_REG</a>	Register 6 of BLOCK7 (KEY3).	0x0114	RO
<a href="#">EFUSE_RD_KEY3_DATA7_REG</a>	Register 7 of BLOCK7 (KEY3).	0x0118	RO
<a href="#">EFUSE_RD_KEY4_DATA0_REG</a>	Register 0 of BLOCK8 (KEY4).	0x011C	RO
<a href="#">EFUSE_RD_KEY4_DATA1_REG</a>	Register 1 of BLOCK8 (KEY4).	0x0120	RO
<a href="#">EFUSE_RD_KEY4_DATA2_REG</a>	Register 2 of BLOCK8 (KEY4).	0x0124	RO
<a href="#">EFUSE_RD_KEY4_DATA3_REG</a>	Register 3 of BLOCK8 (KEY4).	0x0128	RO
<a href="#">EFUSE_RD_KEY4_DATA4_REG</a>	Register 4 of BLOCK8 (KEY4).	0x012C	RO
<a href="#">EFUSE_RD_KEY4_DATA5_REG</a>	Register 5 of BLOCK8 (KEY4).	0x0130	RO
<a href="#">EFUSE_RD_KEY4_DATA6_REG</a>	Register 6 of BLOCK8 (KEY4).	0x0134	RO
<a href="#">EFUSE_RD_KEY4_DATA7_REG</a>	Register 7 of BLOCK8 (KEY4).	0x0138	RO
<a href="#">EFUSE_RD_KEY5_DATA0_REG</a>	Register 0 of BLOCK9 (KEY5).	0x013C	RO
<a href="#">EFUSE_RD_KEY5_DATA1_REG</a>	Register 1 of BLOCK9 (KEY5).	0x0140	RO
<a href="#">EFUSE_RD_KEY5_DATA2_REG</a>	Register 2 of BLOCK9 (KEY5).	0x0144	RO
<a href="#">EFUSE_RD_KEY5_DATA3_REG</a>	Register 3 of BLOCK9 (KEY5).	0x0148	RO
<a href="#">EFUSE_RD_KEY5_DATA4_REG</a>	Register 4 of BLOCK9 (KEY5).	0x014C	RO



Name	Description	Address	Access
<a href="#">EFUSE_RD_KEY5_DATA5_REG</a>	Register 5 of BLOCK9 (KEY5).	0x0150	RO
<a href="#">EFUSE_RD_KEY5_DATA6_REG</a>	Register 6 of BLOCK9 (KEY5).	0x0154	RO
<a href="#">EFUSE_RD_KEY5_DATA7_REG</a>	Register 7 of BLOCK9 (KEY5).	0x0158	RO
<a href="#">EFUSE_RD_SYS_DATA_PART2_0_REG</a>	Register 0 of BLOCK10 (system).	0x015C	RO
<a href="#">EFUSE_RD_SYS_DATA_PART2_1_REG</a>	Register 1 of BLOCK10 (system).	0x0160	RO
<a href="#">EFUSE_RD_SYS_DATA_PART2_2_REG</a>	Register 2 of BLOCK10 (system).	0x0164	RO
<a href="#">EFUSE_RD_SYS_DATA_PART2_3_REG</a>	Register 3 of BLOCK10 (system).	0x0168	RO
<a href="#">EFUSE_RD_SYS_DATA_PART2_4_REG</a>	Register 4 of BLOCK10 (system).	0x016C	RO
<a href="#">EFUSE_RD_SYS_DATA_PART2_5_REG</a>	Register 5 of BLOCK10 (system).	0x0170	RO
<a href="#">EFUSE_RD_SYS_DATA_PART2_6_REG</a>	Register 6 of BLOCK10 (system).	0x0174	RO
<a href="#">EFUSE_RD_SYS_DATA_PART2_7_REG</a>	Register 7 of BLOCK10 (system).	0x0178	RO
<b>Error Status Registers</b>			
<a href="#">EFUSE_RD_REPEAT_ERR0_REG</a>	Programming error record register 0 of BLOCK0.	0x017C	RO
<a href="#">EFUSE_RD_REPEAT_ERR1_REG</a>	Programming error record register 1 of BLOCK0.	0x0180	RO
<a href="#">EFUSE_RD_REPEAT_ERR2_REG</a>	Programming error record register 2 of BLOCK0.	0x0184	RO
<a href="#">EFUSE_RD_REPEAT_ERR3_REG</a>	Programming error record register 3 of BLOCK0.	0x0188	RO
<a href="#">EFUSE_RD_REPEAT_ERR4_REG</a>	Programming error record register 4 of BLOCK0.	0x0190	RO
<a href="#">EFUSE_RD_RS_ERR0_REG</a>	Programming error record register 0 of BLOCK1-10.	0x01C0	RO
<a href="#">EFUSE_RD_RS_ERR1_REG</a>	Programming error record register 1 of BLOCK1-10.	0x01C4	RO
<b>Control/Status Registers</b>			
<a href="#">EFUSE_CLK_REG</a>	eFuse clock configuration register.	0x01C8	R/W
<a href="#">EFUSE_CONF_REG</a>	eFuse operation mode configuration register.	0x01CC	R/W
<a href="#">EFUSE_CMD_REG</a>	eFuse command register.	0x01D4	R/W
<a href="#">EFUSE_DAC_CONF_REG</a>	Controls the eFuse programming voltage.	0x01E8	R/W
<a href="#">EFUSE_STATUS_REG</a>	eFuse status register.	0x01D0	RO
<b>Interrupt Registers</b>			
<a href="#">EFUSE_INT_RAW_REG</a>	eFuse raw interrupt register.	0x01D8	RO
<a href="#">EFUSE_INT_ST_REG</a>	eFuse interrupt status register.	0x01DC	RO
<a href="#">EFUSE_INT_ENA_REG</a>	eFuse interrupt enable register.	0x01E0	R/W
<a href="#">EFUSE_INT_CLR_REG</a>	eFuse interrupt clear register.	0x01E4	WO
<b>Configuration Registers</b>			
<a href="#">EFUSE_RD_TIM_CONF_REG</a>	Configures read timing parameters.	0x01EC	R/W
<a href="#">EFUSE_WR_TIM_CONF0_REG</a>	Configuration register 0 of eFuse programming timing parameters.	0x01F0	R/W
<a href="#">EFUSE_WR_TIM_CONF1_REG</a>	Configuration register 1 of eFuse programming timing parameters.	0x01F4	R/W
<a href="#">EFUSE_WR_TIM_CONF2_REG</a>	Configuration register 2 of eFuse programming timing parameters.	0x01F8	R/W
<b>Version Register</b>			
<a href="#">EFUSE_DATE_REG</a>	Version control register.	0x01FC	R/W



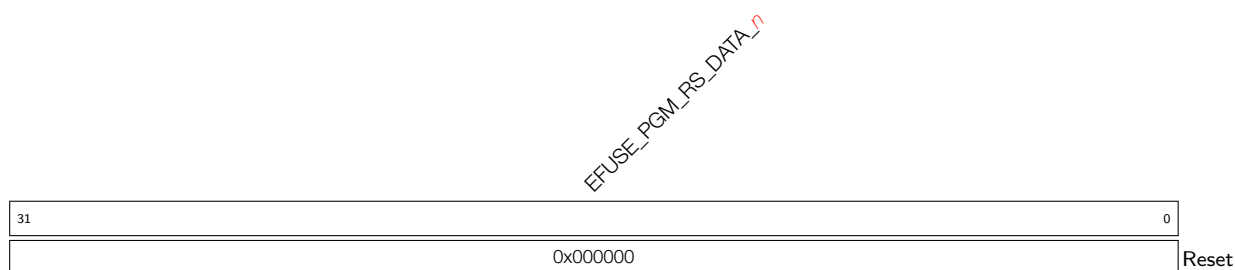
## 16.6 Registers

Register 16.1: EFUSE\_PGM\_DATA<sub>*n*</sub>\_REG (*n*: 0-7) (0x0000+4\**n*)



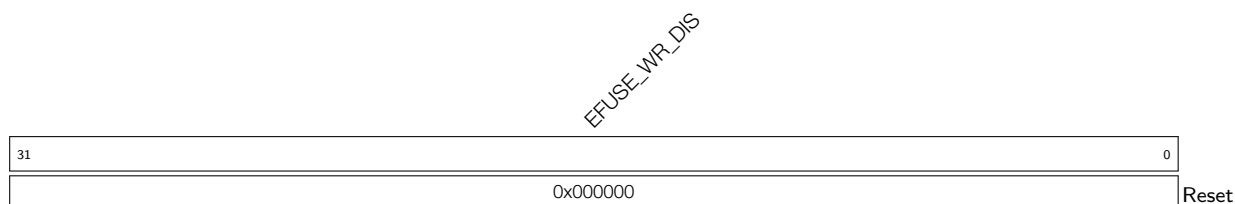
**EFUSE\_PGM\_DATA<sub>*n*</sub>** The content of the *n*th 32-bit data to be programmed. (R/W)

Register 16.2: EFUSE\_PGM\_CHECK\_VALUE<sub>*n*</sub>\_REG (*n*: 0-2) (0x0020+4\**n*)



**EFUSE\_PGM\_RS\_DATA<sub>*n*</sub>** The content of the *n*th 32-bit RS code to be programmed. (R/W)

Register 16.3: EFUSE\_RD\_WR\_DIS\_REG (0x002C)



**EFUSE\_WR\_DIS** Disables programming of individual eFuses. (RO)

Register 16.4: EFUSE\_RD\_REPEAT\_DATA0\_REG (0x0030)

(reserved)			EFUSE_RPT4_RESERVED0			EFUSE_USB_FORCE_NOPERSIST			EFUSE_EXT_PHY_ENABLE			EFUSE_USB_EXCHG_PINS			(reserved)			EFUSE_DIS_DOWNLOAD_MANUAL_ENCRYPT			EFUSE_HARD_DIS_JTAG			EFUSE_SOFT_DIS_JTAG			EFUSE_RPT4_RESERVED5			EFUSE_DIS_BOOT_REMAP			EFUSE_DIS_CAN			EFUSE_DIS_USB			EFUSE_DIS_FORCE_DOWNLOAD			EFUSE_DIS_DOWNLOAD_DCACHE			EFUSE_DIS_DOWNLOAD_ICACHE			EFUSE_DIS_RTC_RAM_BOOT			EFUSE_RD_DIS																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
31	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	Reset																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
0	0	0	0x0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**EFUSE\_RD\_DIS** Disables software reading from individual eFuse blocks (BLOCK4-10). (RO)

**EFUSE\_DIS\_RTC\_RAM\_BOOT** Set this bit to disable boot from RTC RAM. (RO)

**EFUSE\_DIS\_ICACHE** Set this bit to disable ICache. (RO)

**EFUSE\_DIS\_DCACHE** Set this bit to disable DCache. (RO)

**EFUSE\_DIS\_DOWNLOAD\_ICACHE** Disables Icache when SoC is in Download mode. (RO)

**EFUSE\_DIS\_DOWNLOAD\_DCACHE** Disables Dcache when SoC is in Download mode. (RO)

**EFUSE\_DIS\_FORCE\_DOWNLOAD** Set this bit to disable the function that forces chip into download mode. (RO)

**EFUSE\_DIS\_USB** Set this bit to disable USB OTG function. (RO)

**EFUSE\_DIS\_CAN** Set this bit to disable the TWAI Controller function. (RO)

**EFUSE\_DIS\_BOOT\_REMAP** Disables capability to Remap RAM to ROM address space. (RO)

**EFUSE\_RPT4\_RESERVED5** Reserved (used for four backups method). (RO)

**EFUSE\_SOFT\_DIS\_JTAG** Software disables JTAG. When software disabled, JTAG can be activated temporarily by HMAC peripheral. (RO)

**EFUSE\_HARD\_DIS\_JTAG** Hardware disables JTAG permanently. (RO)

**EFUSE\_DIS\_DOWNLOAD\_MANUAL\_ENCRYPT** Disables flash encryption when in download boot modes. (RO)

**EFUSE\_USB\_EXCHG\_PINS** Set this bit to exchange USB D+ and D- pins. (RO)

**EFUSE\_EXT\_PHY\_ENABLE** Set this bit to enable external USB PHY. (RO)

**EFUSE\_USB\_FORCE\_NOPERSIST** If set, forces USB BVALID to 1. (RO)

**EFUSE\_RPT4\_RESERVED0** Reserved (used for four backups method). (RO)

**Register 16.5: EFUSE\_RD\_REPEAT\_DATA1\_REG (0x0034)**

EFUSE_KEY_PURPOSE_1		EFUSE_KEY_PURPOSE_0		EFUSE_SECURE_BOOT_KEY_REVOKE2		EFUSE_SECURE_BOOT_KEY_REVOKE1		EFUSE_SECURE_BOOT_KEY_REVOKE0		EFUSE_SPI_BOOT_CRYPT_CNT		EFUSE_WDT_DELAY_SEL		(reserved)		EFUSE_VDD_SPI_FORCE		EFUSE_VDD_SPI_TIEH		EFUSE_VDD_SPI_XPD		(reserved)	
31	28	27	24	23	22	21	20	18	17	16	15					7	6	5	4	3	0		
0x0		0x0		0	0	0	0x0	0x0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**EFUSE\_VDD\_SPI\_XPD** If VDD\_SPI\_FORCE is 1, this value determines if the VDD\_SPI regulator is powered on. (RO)

**EFUSE\_VDD\_SPI\_TIEH** If VDD\_SPI\_FORCE is 1, determines VDD\_SPI voltage. 0: VDD\_SPI connects to 1.8 V LDO; 1: VDD\_SPI connects to VDD\_RTC\_IO. (RO)

**EFUSE\_VDD\_SPI\_FORCE** Set this bit to use XPD\_VDD\_PSI\_REG and VDD\_SPI\_TIEH to configure VDD\_SPI LDO. (RO)

**EFUSE\_WDT\_DELAY\_SEL** Selects RTC watchdog timeout threshold at startup. 0: 40,000 slow clock cycles; 1: 80,000 slow clock cycles; 2: 160,000 slow clock cycles; 3: 320,000 slow clock cycles. (RO)

**EFUSE\_SPI\_BOOT\_CRYPT\_CNT** Enables encryption and decryption, when an SPI boot mode is set. Feature is enabled 1 or 3 bits are set in the eFuse, disabled otherwise. (RO)

**EFUSE\_SECURE\_BOOT\_KEY\_REVOKE0** If set, revokes use of secure boot key digest 0. (RO)

**EFUSE\_SECURE\_BOOT\_KEY\_REVOKE1** If set, revokes use of secure boot key digest 1. (RO)

**EFUSE\_SECURE\_BOOT\_KEY\_REVOKE2** If set, revokes use of secure boot key digest 2. (RO)

**EFUSE\_KEY\_PURPOSE\_0** Purpose of KEY0. Refer to Table 58 *Key Purpose Values*. (RO)

**EFUSE\_KEY\_PURPOSE\_1** Purpose of KEY1. Refer to Table 58 *Key Purpose Values*. (RO)

**Register 16.6: EFUSE\_RD\_REPEAT\_DATA2\_REG (0x0038)**

EFUSE_FLASH_TPUW				EFUSE_RPT4_RESERVED1				EFUSE_SECURE_BOOT_AGGRESSIVE_REVOKE				EFUSE_SECURE_BOOT_EN				(reserved)				EFUSE_KEY_PURPOSE_5				EFUSE_KEY_PURPOSE_4				EFUSE_KEY_PURPOSE_3				EFUSE_KEY_PURPOSE_2			
31	28	27	22	21	20	19	16	15	12	11	8	7	4	3	0																				
0x0				0x0				0	0	0	0	0	0	0	0	0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	Reset					

Reset

**EFUSE\_KEY\_PURPOSE\_2** Purpose of KEY2. Refer to Table 58 *Key Purpose Values*. (RO)

**EFUSE\_KEY\_PURPOSE\_3** Purpose of KEY3. Refer to Table 58 *Key Purpose Values*. (RO)

**EFUSE\_KEY\_PURPOSE\_4** Purpose of KEY4. Refer to Table 58 *Key Purpose Values*. (RO)

**EFUSE\_KEY\_PURPOSE\_5** Purpose of KEY5. Refer to Table 58 *Key Purpose Values*. (RO)

**EFUSE\_SECURE\_BOOT\_EN** Set this bit to enable secure boot. (RO)

**EFUSE\_SECURE\_BOOT\_AGGRESSIVE\_REVOKE** Set this bit to enable aggressive secure boot key revocation mode. (RO)

**EFUSE\_RPT4\_RESERVED1** Reserved (used for four backups method). (RO)

**EFUSE\_FLASH\_TPUW** Configures flash startup delay after SoC power-up, in unit of (ms/2). When the value is 15, delay is 7.5 ms. (RO)

Register 16.7: EFUSE\_RD\_REPEAT\_DATA3\_REG (0x003C)

EFUSE_RPT4_RESERVED2																			
EFUSE_SECURE_VERSION																			
EFUSE_FORCE_SEND_RESUME																			
EFUSE_FLASH_TYPE																			
EFUSE_PIN_POWER_SELECTION																			
EFUSE_UART_PRINT_SELECTION																			
EFUSE_ENABLE_SECURITY_DOWNLOAD																			
EFUSE_DIS_USB_DOWNLOAD_MODE																			
EFUSE_RPT4_RESERVED3																			
EFUSE_UART_PRINT_CHANNEL																			
EFUSE_DIS_LEGACY_SPI_BOOT																			
EFUSE_DIS_DOWNLOAD_MODE																			
31	27	26					11	10	9	8	7	6	5	4	3	2	1	0	
0x0		0x00							0	0	0	0x0	0	0	0	0	0	0	Reset

**EFUSE\_DIS\_DOWNLOAD\_MODE** Set this bit to disable all download boot modes. (RO)

**EFUSE\_DIS\_LEGACY\_SPI\_BOOT** Set this bit to disable Legacy SPI boot mode. (RO)

**EFUSE\_UART\_PRINT\_CHANNEL** Selects the default UART for printing boot messages. 0: UART0; 1: UART1. (RO)

**EFUSE\_RPT4\_RESERVED3** Reserved (used for four backups method). (RO)

**EFUSE\_DIS\_USB\_DOWNLOAD\_MODE** Set this bit to disable use of USB OTG in UART download boot mode. (RO)

**EFUSE\_ENABLE\_SECURITY\_DOWNLOAD** Set this bit to enable secure UART download mode (read/write flash only). (RO)

**EFUSE\_UART\_PRINT\_CONTROL** Set the default UART boot message output mode.

00: Enabled.

01: Enable when GPIO46 is low at reset.

10: Enable when GPIO46 is high at reset.

11: Disabled. (RO)

**EFUSE\_PIN\_POWER\_SELECTION** Set default power supply for GPIO33-GPIO37, set when SPI flash is initialized. 0: VDD3P3\_CPU; 1: VDD\_SPI. (RO)

**EFUSE\_FLASH\_TYPE** SPI flash type. 0: maximum four data lines, 1: eight data lines. (RO)

**EFUSE\_FORCE\_SEND\_RESUME** If set, forces ROM code to send an SPI flash resume command during SPI boot. (RO)

**EFUSE\_SECURE\_VERSION** Secure version (used by ESP-IDF anti-rollback feature). (RO)

**EFUSE\_RPT4\_RESERVED2** Reserved (used for four backups method). (RO)

**Register 16.8: EFUSE\_RD\_REPEAT\_DATA4\_REG (0x0040)**

(reserved)								EFUSE_RPT4_RESERVED4																								
31								24	23																							0
0	0	0	0	0	0	0	0	0x0000																								Reset

**EFUSE\_RPT4\_RESERVED4** Reserved (used for four backups method). (RO)

**Register 16.9: EFUSE\_RD\_MAC\_SPI\_SYS\_0\_REG (0x0044)**

EFUSE_MAC_0																																
31																															0	
0x000000																																Reset

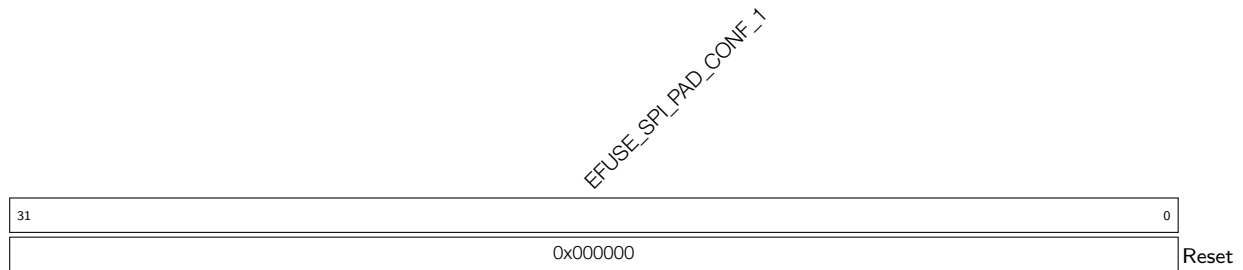
**EFUSE\_MAC\_0** Stores the low 32 bits of MAC address. (RO)

**Register 16.10: EFUSE\_RD\_MAC\_SPI\_SYS\_1\_REG (0x0048)**

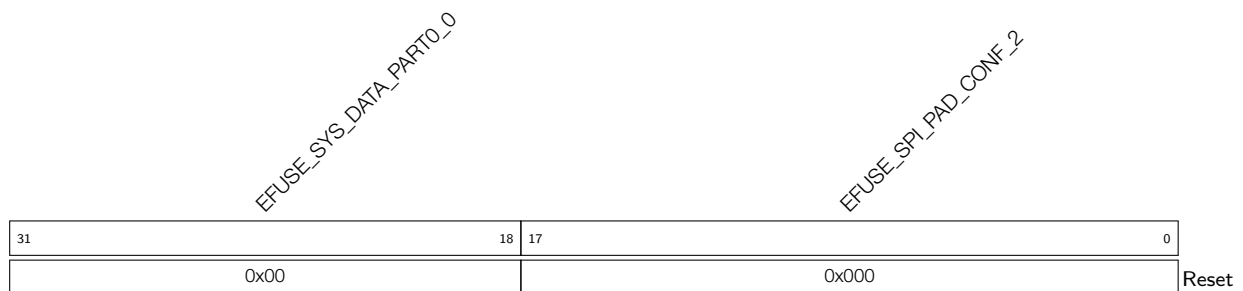
EFUSE_SPI_PAD_CONF_0																EFUSE_MAC_1																
31															16	15															0	
0x00																0x00																Reset

**EFUSE\_MAC\_1** Stores the high 16 bits of MAC address. (RO)

**EFUSE\_SPI\_PAD\_CONF\_0** Stores the zeroth part of SPI\_PAD\_CONF. (RO)

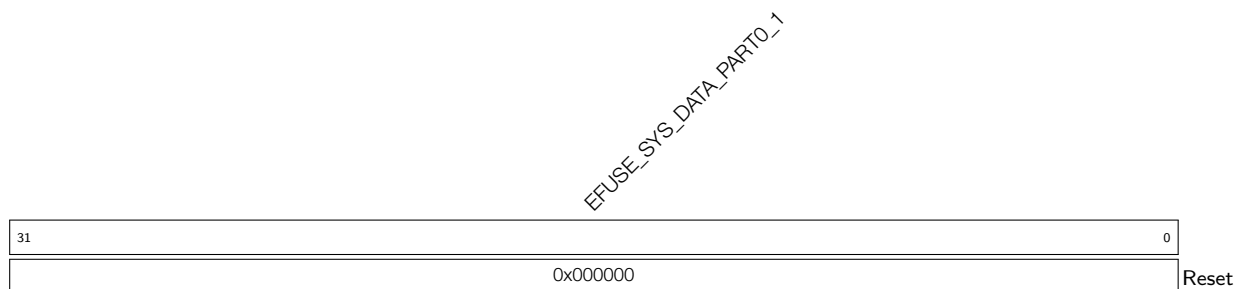
**Register 16.11: EFUSE\_RD\_MAC\_SPI\_SYS\_2\_REG (0x004C)**

**EFUSE\_SPI\_PAD\_CONF\_1** Stores the first part of SPI\_PAD\_CONF. (RO)

**Register 16.12: EFUSE\_RD\_MAC\_SPI\_SYS\_3\_REG (0x0050)**

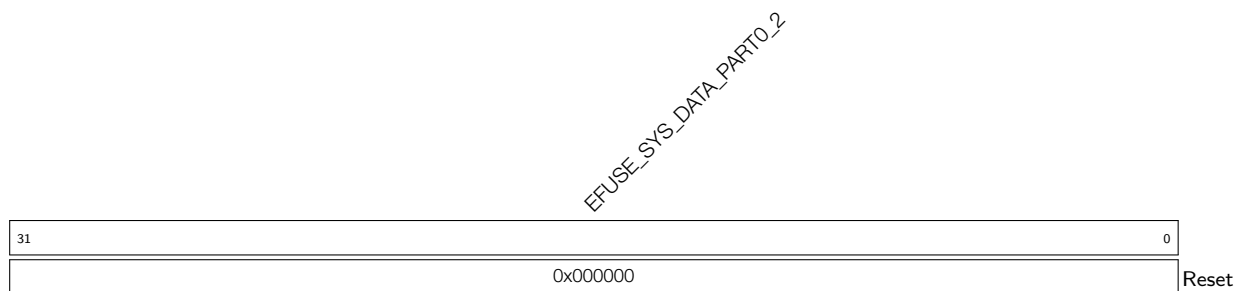
**EFUSE\_SPI\_PAD\_CONF\_2** Stores the second part of SPI\_PAD\_CONF. (RO)

**EFUSE\_SYS\_DATA\_PART0\_0** Stores the zeroth part of the zeroth part of system data. (RO)

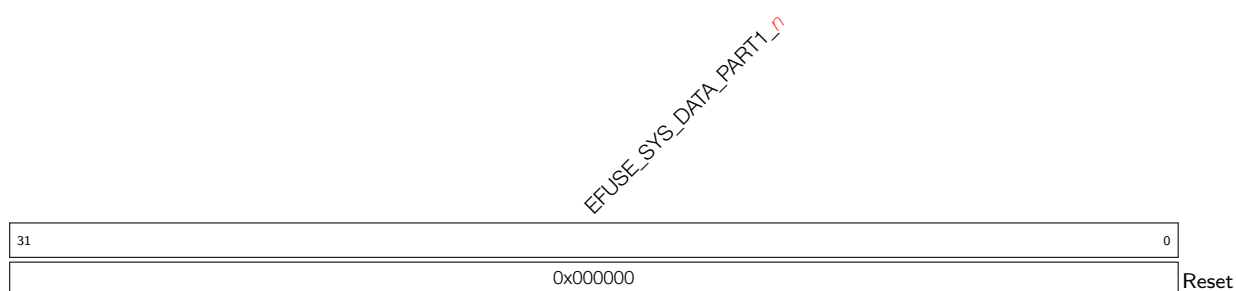
**Register 16.13: EFUSE\_RD\_MAC\_SPI\_SYS\_4\_REG (0x0054)**

**EFUSE\_SYS\_DATA\_PART0\_1** Stores the first part of the zeroth part of system data. (RO)

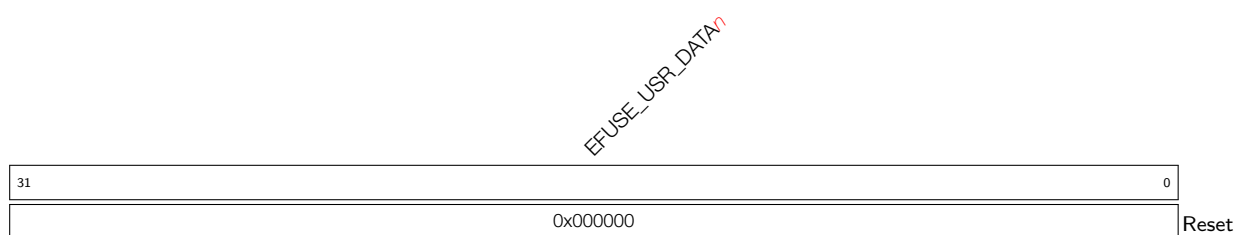
## Register 16.14: EFUSE\_RD\_MAC\_SPI\_SYS\_5\_REG (0x0058)



**EFUSE\_SYS\_DATA\_PART0\_2** Stores the second part of the zeroth part of system data. (RO)

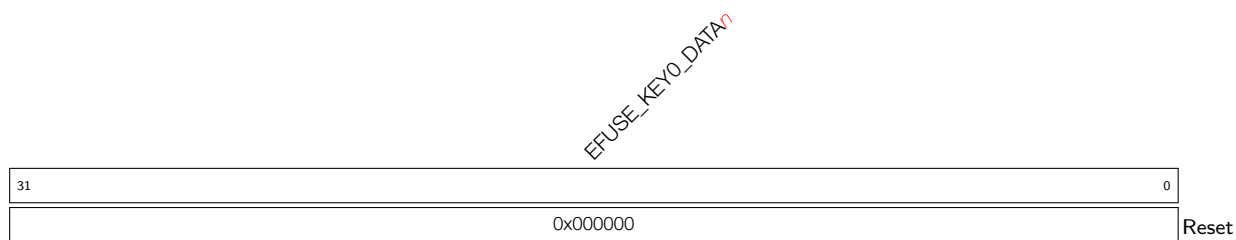
Register 16.15: EFUSE\_RD\_SYS\_DATA\_PART1\_*n*\_REG (*n*: 0-7) (0x005C+4\**n*)

**EFUSE\_SYS\_DATA\_PART1\_*n*** Stores the *n*th 32 bits of the first part of system data. (RO)

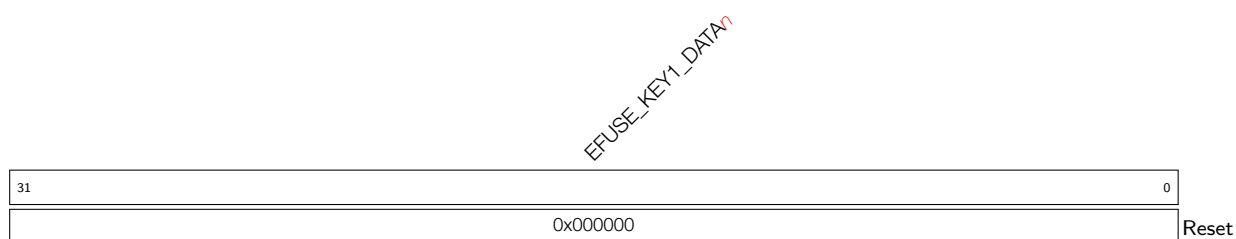
Register 16.16: EFUSE\_RD\_USR\_DATA<sub>*n*</sub>\_REG (*n*: 0-7) (0x007C+4\**n*)

**EFUSE\_USR\_DATA<sub>*n*</sub>** Stores the *n*th 32 bits of BLOCK3 (user). (RO)

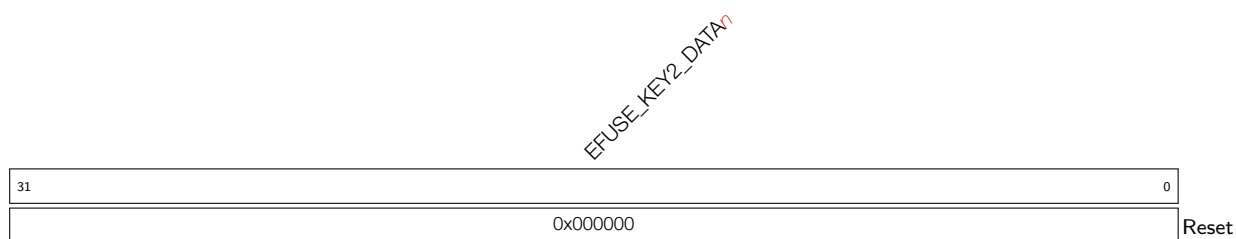


**Register 16.17: EFUSE\_RD\_KEY0\_DATA $n$ \_REG ( $n$ : 0-7) (0x009C+4\* $n$ )**

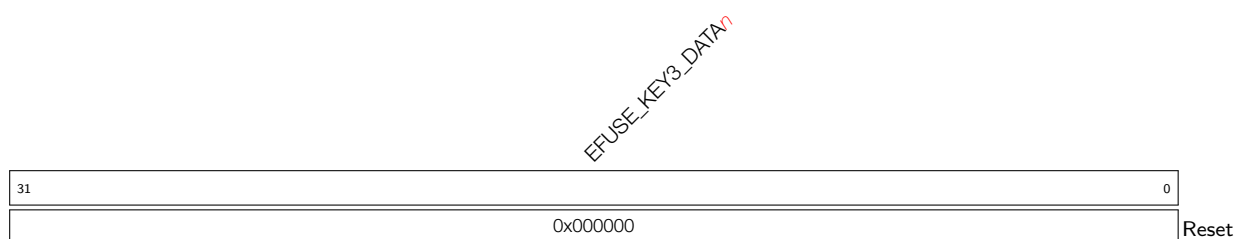
**EFUSE\_KEY0\_DATA $n$**  Stores the  $n$ th 32 bits of KEY0. (RO)

**Register 16.18: EFUSE\_RD\_KEY1\_DATA $n$ \_REG ( $n$ : 0-7) (0x00BC+4\* $n$ )**

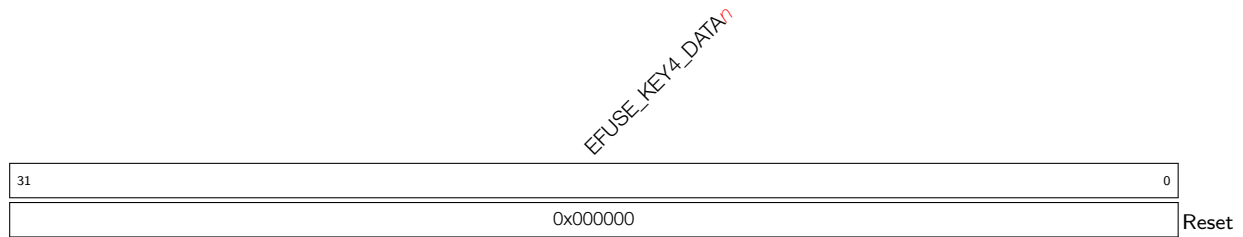
**EFUSE\_KEY1\_DATA $n$**  Stores the  $n$ th 32 bits of KEY1. (RO)

**Register 16.19: EFUSE\_RD\_KEY2\_DATA $n$ \_REG ( $n$ : 0-7) (0x00DC+4\* $n$ )**

**EFUSE\_KEY2\_DATA $n$**  Stores the  $n$ th 32 bits of KEY2. (RO)

**Register 16.20: EFUSE\_RD\_KEY3\_DATA $n$ \_REG ( $n$ : 0-7) (0x00FC+4\* $n$ )**

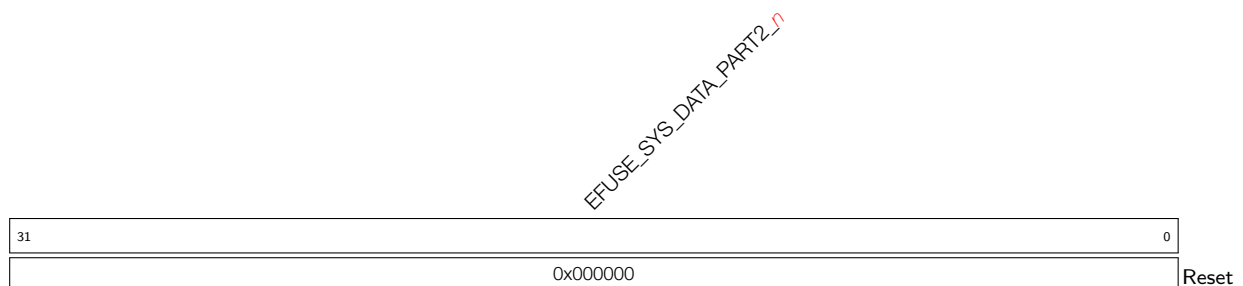
**EFUSE\_KEY3\_DATA $n$**  Stores the  $n$ th 32 bits of KEY3. (RO)

**Register 16.21: EFUSE\_RD\_KEY4\_DATA<sub>*n*</sub>\_REG (*n*: 0-7) (0x011C+4\**n*)**

**EFUSE\_KEY4\_DATA<sub>*n*</sub>** Stores the *n*th 32 bits of KEY4. (RO)

**Register 16.22: EFUSE\_RD\_KEY5\_DATA<sub>*n*</sub>\_REG (*n*: 0-7) (0x013C+4\**n*)**

**EFUSE\_KEY5\_DATA<sub>*n*</sub>** Stores the *n*th 32 bits of KEY5. (RO)

**Register 16.23: EFUSE\_RD\_SYS\_DATA\_PART2\_<sub>*n*</sub>\_REG (*n*: 0-7) (0x015C+4\**n*)**

**EFUSE\_SYS\_DATA\_PART2\_<sub>*n*</sub>** Stores the *n*th 32 bits of the 2nd part of system data. (RO)

Register 16.24: EFUSE\_RD\_REPEAT\_ERR0\_REG (0x017C)

(reserved)			EFUSE_RPT4_RESERVED_ERR			EFUSE_USB_FORCE_NOPERSIST_ERR			EFUSE_EXT_PHY_ENABLE_ERR			EFUSE_USB_EXCHG_PINS_ERR			(reserved)			EFUSE_DIS_DOWNLOAD_MANUAL_ENCRYPT_ERR			EFUSE_DIS_DOWNLOAD_ICACHE_ERR			EFUSE_DIS_DOWNLOAD_DCACHE_ERR			EFUSE_RD_DIS_ERR			
31	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
0	0	0	0x0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

**EFUSE\_RD\_DIS\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_RD\\_DIS](#). (RO)

**EFUSE\_DIS\_RTC\_RAM\_BOOT\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_RTC\\_RAM\\_BOOT](#). (RO)

**EFUSE\_DIS\_ICACHE\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_ICACHE](#). (RO)

**EFUSE\_DIS\_DCACHE\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_DCACHE](#). (RO)

**EFUSE\_DIS\_DOWNLOAD\_ICACHE\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_DOWNLOAD\\_ICACHE](#). (RO)

**EFUSE\_DIS\_DOWNLOAD\_DCACHE\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_DOWNLOAD\\_DCACHE](#). (RO)

**EFUSE\_DIS\_FORCE\_DOWNLOAD\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_FORCE\\_DOWNLOAD](#). (RO)

**EFUSE\_DIS\_USB\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_USB](#). (RO)

**EFUSE\_DIS\_CAN\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_CAN](#). (RO)

**EFUSE\_DIS\_BOOT\_REMAP\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_BOOT\\_REMAP](#). (RO)

**EFUSE\_RPT4\_RESERVED5\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_RPT4\\_RESERVED5](#). (RO)

**EFUSE\_SOFT\_DIS\_JTAG\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_SOFT\\_DIS\\_JTAG](#). (RO)

**EFUSE\_HARD\_DIS\_JTAG\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_HARD\\_DIS\\_JTAG](#). (RO)

Continued on the next page...

**Register 16.24: EFUSE\_RD\_REPEAT\_ERR0\_REG (0x017C)**

Continued from the previous page...

**EFUSE\_DIS\_DOWNLOAD\_MANUAL\_ENCRYPT\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_DOWNLOAD\\_MANUAL\\_ENCRYPT](#). (RO)

**EFUSE\_USB\_EXCHG\_PINS\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_USB\\_EXCHG\\_PINS](#). (RO)

**EFUSE\_EXT\_PHY\_ENABLE\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_EXT\\_PHY\\_ENABLE](#). (RO)

**EFUSE\_USB\_FORCE\_NOPERSIST\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_USB\\_FORCE\\_NOPERSIST](#). (RO)

**EFUSE\_RPT4\_RESERVED0\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_RPT4\\_RESERVED0](#). (RO)

## 365

ESP32-S2 TRM (Preliminary V0.5)

**EFUSE\_KEY\_PURPOSE\_1\_ERR** Any bit equal to 1 denotes a programming error in EFUSE KEY PURPOSE 1. (RO)

**Register 16.26: EFUSE\_RD\_REPEAT\_ERR2\_REG (0x0184)**

EFUSE_FLASH_TPUW_ERR				EFUSE_RPT4_RESERVED1_ERR				EFUSE_SECURE_BOOT_AGGRESSIVE_REVOKE_ERR				EFUSE_SECURE_BOOT_EN_ERR				(reserved)				EFUSE_KEY_PURPOSE_5_ERR				EFUSE_KEY_PURPOSE_4_ERR				EFUSE_KEY_PURPOSE_3_ERR				EFUSE_KEY_PURPOSE_2_ERR			
31	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						
0x0				0x0				0	0	0	0	0	0	0	0x0				0x0				0x0				0x0				Reset				

**EFUSE\_KEY\_PURPOSE\_2\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_KEY\\_PURPOSE\\_2](#). (RO)

**EFUSE\_KEY\_PURPOSE\_3\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_KEY\\_PURPOSE\\_3](#). (RO)

**EFUSE\_KEY\_PURPOSE\_4\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_KEY\\_PURPOSE\\_4](#). (RO)

**EFUSE\_KEY\_PURPOSE\_5\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_KEY\\_PURPOSE\\_5](#). (RO)

**EFUSE\_SECURE\_BOOT\_EN\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_SECURE\\_BOOT\\_EN](#). (RO)

**EFUSE\_SECURE\_BOOT\_AGGRESSIVE\_REVOKE\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_SECURE\\_BOOT\\_AGGRESSIVE\\_REVOKE](#). (RO)

**EFUSE\_RPT4\_RESERVED1\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_RPT4\\_RESERVED1](#). (RO)

**EFUSE\_FLASH\_TPUW\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_FLASH\\_TPUW](#). (RO)

Register 16.27: EFUSE\_RD\_REPEAT\_ERR3\_REG (0x0188)

EFUSE_RPT4_RESERVED2_ERR																												
EFUSE_SECURE_VERSION_ERR																												
EFUSE_FORCE_SEND_RESUME_ERR																												
EFUSE_FLASH_TYPE_ERR																												
EFUSE_PIN_POWER_SELECTION_ERR																												
EFUSE_UART_PRINT_CONTROL_ERR																												
EFUSE_ENABLE_SECURITY_DOWNLOAD_ERR																												
EFUSE_DIS_USB_DOWNLOAD_MODE_ERR																												
EFUSE_RPT4_RESERVED3_ERR																												
EFUSE_UART_PRINT_CHANNEL_ERR																												
EFUSE_DIS_LEGACY_SPI_BOOT_ERR																												
EFUSE_DIS_DOWNLOAD_MODE_ERR																												
31		27		26												11		10	9	8	7	6	5	4	3	2	1	0
0x0		0x00										0		0	0	0x0	0	0	0	0	0	0	0	0	0	0	0	Reset

**EFUSE\_DIS\_DOWNLOAD\_MODE\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_DOWNLOAD\\_MODE](#). (RO)

**EFUSE\_DIS\_LEGACY\_SPI\_BOOT\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_LEGACY\\_SPI\\_BOOT](#). (RO)

**EFUSE\_UART\_PRINT\_CHANNEL\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_UART\\_PRINT\\_CHANNEL](#). (RO)

**EFUSE\_RPT4\_RESERVED3\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_RPT4\\_RESERVED3](#). (RO)

**EFUSE\_DIS\_USB\_DOWNLOAD\_MODE\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_DIS\\_USB\\_DOWNLOAD\\_MODE](#). (RO)

**EFUSE\_ENABLE\_SECURITY\_DOWNLOAD\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_ENABLE\\_SECURITY\\_DOWNLOAD](#). (RO)

**EFUSE\_UART\_PRINT\_CONTROL\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_UART\\_PRINT\\_CONTROL](#). (RO)

**EFUSE\_PIN\_POWER\_SELECTION\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_PIN\\_POWER\\_SELECTION](#). (RO)

**EFUSE\_FLASH\_TYPE\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_FLASH\\_TYPE](#). (RO)

**EFUSE\_FORCE\_SEND\_RESUME\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_FORCE\\_SEND\\_RESUME](#). (RO)

**EFUSE\_SECURE\_VERSION\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_SECURE\\_VERSION](#). (RO)

**EFUSE\_RPT4\_RESERVED2\_ERR** Any bit equal to 1 denotes a programming error in [EFUSE\\_RPT4\\_RESERVED2](#). (RO)

### Register 16.28: EFUSE\_RD\_REPEAT\_ERR4\_REG (0x0190)

Diagram illustrating the structure of the `EFUSE_RPT4_RESERVED4_ERR` register:

- Bits 31 to 24: (reserved)
- Bits 23 to 0: Error code field, initialized to `0x0000`.
- Reset state is indicated.

**EFUSE\_RPT4\_RESERVED4\_ERR** If any bit in RPT4\_RESERVED4 is 1, there is a programming error in [EFUSE\\_RPT4\\_RESERVED4](#). (RO)

### Register 16.29: EFUSE\_RD\_RS\_ERR0\_REG (0x01C0)

[illegible]

**EFUSE\_MAC\_SPI\_8M\_ERR\_NUM** The value of this signal means the number of error bytes in BLOCK1. (RO)

**EFUSE\_MAC\_SPI\_8M\_FAIL** 0: Means no failure and that the data of BLOCK1 is reliable; 1: Means that programming BLOCK1 data failed and the number of error bytes is over 5. (RO)

**EFUSE\_SYS\_PART1\_NUM** The value of this signal means the number of error bytes in BLOCK2.  
(RO)

**EFUSE\_SYS\_PART1\_FAIL** 0: Means no failure and that the data of BLOCK2 is reliable; 1: Means that programming BLOCK2 data failed and the number of error bytes is over 5. (RO)

**EFUSE\_USR\_DATA\_ERR\_NUM** The value of this signal means the number of error bytes in BLOCK3.  
(RO)

**EFUSE\_USR\_DATA\_FAIL** 0: Means no failure and that the data of BLOCK3 is reliable; 1: Means that programming BLOCK3 data failed and the number of error bytes is over 5. (RO)

**EFUSE\_KEY<sub>n</sub>\_ERR\_NUM** The value of this signal means the number of error bytes in KEY<sub>n</sub>. (RO)

**EFUSE\_KEY $n$ \_FAIL** 0: Means no failure and that the data of KEY $n$  is reliable; 1: Means that programming KEY $n$  failed and the number of error bytes is over 5. (RO)



Register 16.30: EFUSE\_RD\_RS\_ERR1\_REG (0x01C4)

(reserved)																								EFUSE_SYS_PART2_FAIL		EFUSE_SYS_PART2_ERR_NUM		EFUSE_KEY5_FAIL		EFUSE_KEY5_ERR_NUM			
31																								8	7	6	4		3	2	0		Reset
0 0																								0	0x0		0	0x0					

**EFUSE\_KEY5\_ERR\_NUM** The value of this signal means the number of error bytes in KEY5. (RO)

**EFUSE\_KEY5\_FAIL** 0: Means no failure and that the data of KEY5 is reliable; 1: Means that programming user data failed and the number of error bytes is over 5. (RO)

**EFUSE\_SYS\_PART2\_ERR\_NUM** The value of this signal means the number of error bytes in BLOCK10. (RO)

**EFUSE\_SYS\_PART2\_FAIL** 0: Means no failure and that the data of BLOCK10 is reliable; 1: Means that programming BLOCK10 data failed and the number of error bytes is over 5. (RO)

Register 16.31: EFUSE\_CLK\_REG (0x01C8)

(reserved)																EFUSE_CLK_EN										(reserved)										EFUSE_EFUSE_MEM_FORCE_PU				EFUSE_MEM_CLK_FORCE_ON				EFUSE_EFUSE_MEM_FORCE_PD									
31																17										16		15										3										2		1		0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0										0		1		0		Reset																			

**EFUSE\_EFUSE\_MEM\_FORCE\_PD** If set, forces eFuse SRAM into power-saving mode. (R/W)

**EFUSE\_MEM\_CLK\_FORCE\_ON** If set, forces to activate clock signal of eFuse SRAM. (R/W)

**EFUSE\_EFUSE\_MEM\_FORCE\_PU** If set, forces eFuse SRAM into working mode. (R/W)

**EFUSE\_CLK\_EN** If set, forces to enable clock signal of eFuse memory. (R/W)

**Register 16.32: EFUSE\_CONF\_REG (0x01CC)**

(reserved)																EFUSE_OP_CODE																															
31																15																0															
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x00																Reset															

**EFUSE\_OP\_CODE** 0x5A5A: Operate programming command; 0x5AA5: Operate read command.  
(R/W)

**Register 16.33: EFUSE\_CMD\_REG (0x01D4)**

(reserved)																								EFUSE_BLK_NUM				EFUSE_PGM_CMD		EFUSE_READ_CMD	
31																								6	5	2		1	0		
0 0																								0x0		0		0		Reset	

**EFUSE\_READ\_CMD** Set this bit to send read command. (R/W)

**EFUSE\_PGM\_CMD** Set this bit to send programming command. (R/W)

**EFUSE\_BLK\_NUM** The serial number of the block to be programmed. Value 0-10 corresponds to block number 0-10, respectively. (R/W)

**Register 16.34: EFUSE\_DAC\_CONF\_REG (0x01E8)**

(reserved)																EFUSE_OE_CLR				EFUSE_DAC_NUM				EFUSE_DAC_CLK_PAD_SEL				EFUSE_DAC_CLK_DIV																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
31																18				17	16																9				8	7																0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**EFUSE\_DAC\_CLK\_DIV** Controls the division factor of the rising clock of the programming voltage.  
(R/W)

**EFUSE\_DAC\_CLK\_PAD\_SEL** Don't care. (R/W)

**EFUSE\_DAC\_NUM** Controls the rising period of the programming voltage. (R/W)

**EFUSE\_OE\_CLR** Reduces the power supply of the programming voltage. (R/W)

**Register 16.35: EFUSE\_STATUS\_REG (0x01D0)**

(reserved)																EFUSE_REPEAT_ERR_CNT										(reserved)								EFUSE_STATE																																																																																																																																																																																																																																																																													
31																18																17																10																9																4																3																0																																																																																																																																																																																															
0																0																0																0																0																0																0																0																0																0																0x0																0																0																0																0																0																0																0x0																Reset															

**EFUSE\_STATE** Indicates the state of the eFuse state machine. (RO)

**EFUSE\_REPEAT\_ERR\_CNT** Indicates the number of error bits during programming BLOCK0. (RO)

**Register 16.36: EFUSE\_INT\_RAW\_REG (0x01D8)**

(reserved)																															EFUSE_PGM_DONE_INT_RAW EFUSE_READ_DONE_INT_RAW																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
31																															2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**EFUSE\_READ\_DONE\_INT\_RAW** The raw bit signal for read\_done interrupt. (RO)

**EFUSE\_PGM\_DONE\_INT\_RAW** The raw bit signal for pgm\_done interrupt. (RO)

**Register 16.37: EFUSE\_INT\_ST\_REG (0x01DC)**

(reserved)																															EFUSE_PGM_DONE_INT_ST EFUSE_READ_DONE_INT_ST				
31																															2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**EFUSE\_READ\_DONE\_INT\_ST** The status signal for read\_done interrupt. (RO)

**EFUSE\_PGM\_DONE\_INT\_ST** The status signal for pgm\_done interrupt. (RO)

**Register 16.38: EFUSE\_INT\_ENA\_REG (0x01E0)**

(reserved)																															EFUSE_PGM_DONE_INT_ENA EFUSE_READ_DONE_INT_ENA																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
31																															2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**EFUSE\_READ\_DONE\_INT\_ENA** The enable signal for read\_done interrupt. (R/W)

**EFUSE\_PGM\_DONE\_INT\_ENA** The enable signal for pgm\_done interrupt. (R/W)

**Register 16.39: EFUSE\_INT\_CLR\_REG (0x01E4)**

(reserved)																															EFUSE_PGM_DONE_INT_CLR EFUSE_READ_DONE_INT_CLR																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
31																															2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**EFUSE\_READ\_DONE\_INT\_CLR** The clear signal for read\_done interrupt. (WO)

**EFUSE\_PGM\_DONE\_INT\_CLR** The clear signal for pgm\_done interrupt. (WO)

**Register 16.40: EFUSE\_RD\_TIM\_CONF\_REG (0x01EC)**

EFUSE_READ_INIT_NUM								EFUSE_TSUR_A								EFUSE_TRD								EFUSE_THR_A								
31	24				23	16				15	8				7	0																
0x12								0x1								0x1								0x1								Reset

**EFUSE\_THR\_A** Configures the hold time of read operation. (R/W)

**EFUSE\_TRD** Configures the length of pulse of read operation. (R/W)

**EFUSE\_TSUR\_A** Configures the setup time of read operation. (R/W)

**EFUSE\_READ\_INIT\_NUM** Configures the initial read time of eFuse. (R/W)

**Register 16.41: EFUSE\_WR\_TIM\_CONF0\_REG (0x01F0)**

<div>EFUSE_TPGM</div>																<div>EFUSE_TPGM_INACTIVE</div>								<div>EFUSE_THP_A</div>									
31																16	15	8							7	0							
0xc8																0x1								0x1								Reset	

**EFUSE\_THP\_A** Configures the hold time of programming operation. (R/W)

**EFUSE\_TPGM\_INACTIVE** Configures the length of pulse during programming 0 to eFuse. (R/W)

**EFUSE\_TPGM** Configures the length of pulse during programming 1 to eFuse. (R/W)

**Register 16.42: EFUSE\_WR\_TIM\_CONF1\_REG (0x01F4)**

<div>(reserved)</div>																<div>EFUSE_PWR_ON_NUM</div>																<div>EFUSE_TSUP_A</div>																																															
31								24								23								8								7								0																																							
0								0								0								0								0								0								0								0x2880								0x1								Reset							

**EFUSE\_TSUP\_A** Configures the setup time of programming operation. (R/W)

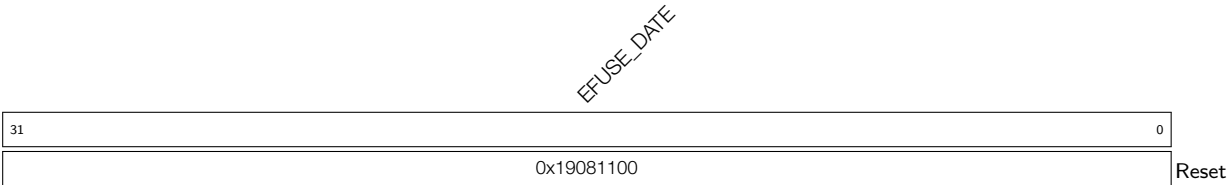
**EFUSE\_PWR\_ON\_NUM** Configures the power up time for VDDQ. (R/W)

**Register 16.43: EFUSE\_WR\_TIM\_CONF2\_REG (0x01F8)**

(reserved)																EFUSE_PWR_OFF_NUM																																															
31																16																15																0															
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x190																Reset																															

**EFUSE\_PWR\_OFF\_NUM** Configures the power outage time for VDDQ. (R/W)

Register 16.44: EFUSE\_DATE\_REG (0x01FC)



**EFUSE\_DATE** Version control register. (R/W)

## 17. I2C Controller

### 17.1 Overview

The I2C (Inter-Integrated Circuit) controller allows ESP32-S2 to communicate with multiple peripheral devices. These peripheral devices can share one bus.

### 17.2 Features

The I2C controller has the following features:

- Master mode and slave mode
- Multi-master and multi-slave communication
- Standard mode (100 kbit/s)
- Fast mode (400 kbit/s)
- 7-bit addressing and 10-bit addressing
- Continuous data transfer in master mode achieved by pulling SCL low
- Programmable digital noise filtering
- Double addressing mode

### 17.3 I2C Functional Description

#### 17.3.1 I2C Introduction

The I2C bus has two lines, namely a serial data line (SDA) and a serial clock line (SCL). Both SDA and SCL lines are open-drain. The I2C bus is connected to multiple devices, usually a single or multiple masters and a single or multiple slaves. However, only one master device can access a slave at a time.

The master initiates communication by generating a start condition: pulling the SDA line low while SCL is high, and sending nine clock pulses via SCL. The first eight pulses are used to transmit a byte, which consists of a 7-bit address followed by a read/write ( $R/\overline{W}$ ) bit. If the address of a slave matches the 7-bit address transmitted, this matching slave can respond by pulling SDA low on the ninth clock pulse. The master and the slave can send or receive data according to the  $R/\overline{W}$  bit. Whether to terminate the data transfer or not is determined by the logic level of the acknowledge (ACK) bit. During data transfer, SDA changes only when SCL is low. Once finishing communication, the master sends a STOP condition: pulling SDA up while SCL is high. If a master both reads and writes data in one transfer, then it should send a RESTART condition, a slave address and a  $R/\overline{W}$  bit before changing its operation.

### 17.3.2 I2C Architecture

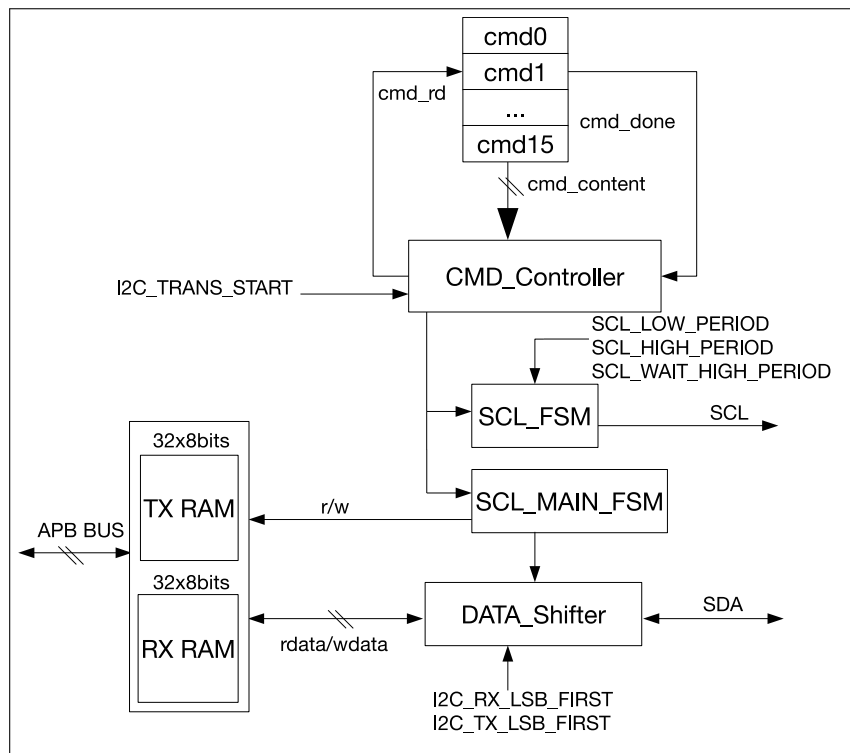


Figure 17-1. I2C Master Architecture

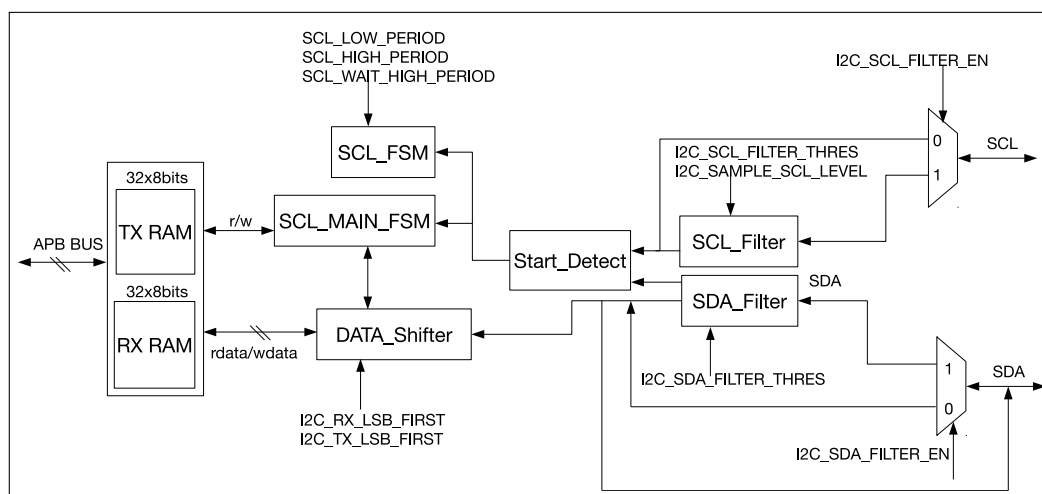


Figure 17-2. I2C Slave Architecture

The I2C controller runs either in master mode or slave mode, which is determined by `I2C_MS_MODE` bit. Figure 17-1 shows the architecture of a master, while Figure 17-2 shows that of a slave. The I2C controller has the following main parts: transmit and receive memory (TX/RX RAM), command controller (CMD\_Controller), SCL clock controller (SCL\_FSM), SDA data controller (SCL\_MAIN\_FSM), serial-to-parallel data converter (DATA\_Shifter), filter for SCL (SCL\_Filter) and filter for SDA (SDA\_Filter).



### 17.3.2.1 TX/RX RAM

Both TX RAM and RX RAM are  $32 \times 8$  bits. TX RAM stores data that the I2C controller needs to send. During communication, when the I2C controller needs to send data (except acknowledgement bits), it reads data from TX RAM and sends it sequentially via SDA. When the I2C controller works in master mode, all data must be stored in TX RAM in the order they will be sent to slaves. The data stored in TX RAM include slave addresses, read/write bits, register addresses (only in dual addressing mode) and data to be sent. When the I2C controller works in slave mode, TX RAM only stores data to be sent.

RX RAM stores data the I2C controller receives during communication. When the I2C controller works in slave mode, neither slave addresses sent by the master nor register addresses (only in double addressing mode) will be stored into RX RAM. Values of RX RAM can be read by software after I2C communication completes.

Both TX RAM and RX RAM can be accessed in FIFO or non-FIFO mode. The `I2C_NONFIFO_EN` bit sets FIFO or non-FIFO mode.

TX RAM can be read and written by the CPU. The CPU writes to TX RAM either in FIFO mode or in non-FIFO mode (direct address). In FIFO mode, the CPU writes to TX RAM via fixed address `I2C_DATA_REG`, with addresses for writing in TX RAM incremented automatically by hardware. In non-FIFO mode, the CPU accesses TX RAM directly via address fields (`I2C Base Address` + 0x100) ~ (`I2C Base Address` + 0x17C). Each byte in TX RAM occupies an entire word in the address space. Therefore, the address of the first byte is `I2C Base Address` + 0x100, the second byte `I2C Base Address` + 0x104, the third byte `I2C Base Address` + 0x108, and so on. The CPU can only read TX RAM via direct addresses. Unlike addresses for writing, TX RAM must be read back from addresses starting at `I2C Base Address` + 0x80.

RX RAM can only be read by the CPU. The CPU reads RX RAM either in FIFO mode or in non-FIFO mode (direct address). In FIFO mode, the CPU reads RX RAM via fixed address `I2C_DATA_REG`, with addresses for reading RX RAM incremented automatically by hardware. In non-FIFO mode, the CPU accesses TX RAM directly via address fields (`I2C Base Address` + 0x100) ~ (`I2C Base Address` + 0x17C). Each byte in RX RAM occupies an entire word in the address space. Therefore, the address of the first byte is `I2C Base Address` + 0x100, the second byte `I2C Base Address` + 0x104, the third byte `I2C Base Address` + 0x108 and so on.

Given that addresses for writing to TX RAM have an identical range with those for reading RX RAM, TX RAM and RX RAM can be seen as a  $32 \times 8$  RAM. In following sections TX RAM and RX RAM are referred to as RAM.

### 17.3.2.2 CMD\_Controller

] When the I2C controller works in master mode, the integrated CMD\_Controller module reads commands from 16 sequential command registers and controls SCL\_FSM and SDA\_FSM accordingly.

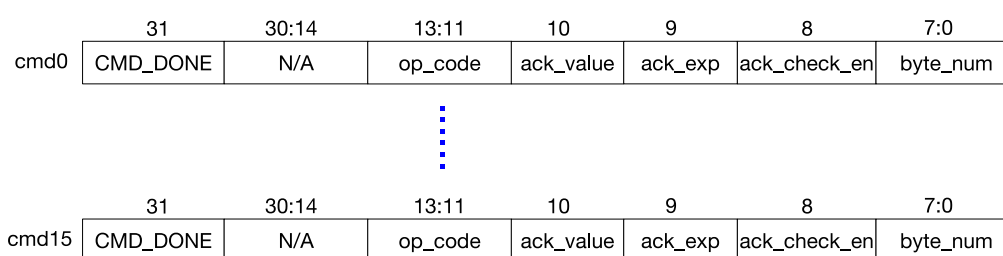


Figure 17-3. Structure of I2C Command Register

Command registers, whose structure is illustrated in Figure 17-3, are active only when the I2C controller works in

master mode. Fields of command registers are:

1. CMD\_DONE: Indicates that a command has been executed. After each command has been executed, the corresponding CMD\_DONE bit is set to 1 by hardware. By reading this bit, software can tell if the command has been executed. When writing new commands, this bit must be cleared by software.
2. op\_code: Indicates the command. The I2C controller supports five commands:
  - RSTART: op\_code = 0: The I2C controller sends a START bit and a RESTART bit defined by the I2C protocol.
  - WRITE: op\_code = 1: The I2C controller sends a slave address, a register address (only in double addressing mode) and data to the slave.
  - READ: op\_code = 2: The I2C controller reads data from the slave.
  - STOP: op\_code = 3: The I2C controller sends a STOP bit defined by the I2C protocol. This code also indicates that the command sequence has been executed, and the CMD\_Controller stops reading commands. After restarted by software, the CMD\_Controller resumes reading commands from command register 0.
  - END: op\_code = 4: The I2C controller pulls the SCL pin down and suspends I2C communication. This code also indicates that the command sequence has completed, and the CMD\_Controller stops executing commands. Once software refreshes data in command registers and the RAM, the CMD\_Controller can be restarted to execute commands from command register 0 again.
3. ack\_value: Used to configure the level of the ACK bit sent by the I2C controller during a read operation. This bit is ignored during RSTART, STOP, END and WRITE conditions.
4. ack\_exp: Used to configure the level of the ACK bit expected by the I2C controller during a write operation. This bit is ignored during RSTART, STOP, END and READ conditions.
5. ack\_check\_en: Used to enable the I2C controller during a write operation to check whether ACK's level sent by the slave matches ack\_exp in the command. If this bit is set and the level received does not match ack\_exp in the WRITE command, the master will generate an I2C\_NACK\_INT interrupt and a STOP condition for data transfer. If this bit is cleared, the controller will not check the ACK level sent by the slave. This bit is ignored during RSTART, STOP, END and READ conditions.
6. byte\_num: Specifies the length of data (in bytes) to be read or written. Can range from 1 to 255 bytes. This bit is ignored during RSTART, STOP and END conditions.

Each command sequence is executed starting from command register 0 and terminated by a STOP or an END. Therefore, all 16 command registers must have a STOP or an END command.

A complete data transfer on the I2C bus should be initiated by a START and terminated by a STOP. The transfer process may be completed using multiple sequences, separated by END commands. Each sequence may differ in the direction of data transfer, clock frequency, slave addresses, data length, data to be transmitted, etc. This allows efficient use of available peripheral RAM and also achieves more flexible I2C communication.

### 17.3.2.3 SCL\_FSM

The integrated SCL\_FSM module controls the SCL clock line. The frequency and duty cycle of SCL is configured using [I2C\\_SCL\\_LOW\\_PERIOD\\_REG](#), [I2C\\_SCL\\_HIGH\\_PERIOD\\_REG](#) and [I2C\\_SCL\\_WAIT\\_HIGH\\_PERIOD](#). After

being in non-IDLE state for over `I2C_SCL_ST_TO` clock cycles, SCL\_FSM triggers an `I2C_SCL_ST_TO_INT` interrupt and returns to IDLE state.

#### 17.3.2.4 SCL\_MAIN\_FSM

The integrated SCL\_MAIN\_FSM module controls the SDA data line and data storage. After being in non-IDLE state for over `I2C_SCL_MAIN_ST_TO` clock cycles, SCL\_MAIN\_FSM triggers an `I2C_SCL_MAIN_ST_TO_INT` interrupt and returns to IDLE state.

#### 17.3.2.5 DATA\_Shifter

The integrated DATA\_Shifter module is used for serial/parallel conversion, converting TX RAM byte data to an outgoing serial bitstream or an incoming serial bitstream to RX RAM byte data. `I2C_RX_LSB_FIRST` and `I2C_TX_LSB_FIRST` can be used to select LSB- or MSB-first storage and transmission of data.

#### 17.3.2.6 SCL\_Filter and SDA\_Filter

The integrated SCL\_Filter and SDA\_Filter modules are identical and are used to filter signal noises on SCL and SDA, respectively. These filters can be enabled or disabled by configuring `I2C_SCL_FILTER_EN` and `I2C_SDA_FILTER_EN`.

SCL\_Filter samples input signals on the SCL line continuously. These input signals are valid only if they remain unchanged for consecutive `I2C_SCL_FILTER_THRES` clock cycles. Given that only valid input signals can pass through the filter, SCL\_Filter can remove glitches whose pulse width is lower than `I2C_SCL_FILTER_THRES` APB clock cycles.

SDA\_Filter is identical to SCL\_Filter, only applied to the SDA line. The threshold pulse width is provided in the `I2C_SDA_FILTER_THRES` register.

### 17.3.3 I2C Bus Timing

The I2C controller may use APB\_CLK or REF\_TICK as its clock source. When `I2C_REF_ALWAYS_ON` is 1, APB\_CLK is used; when `I2C_REF_ALWAYS_ON` is 0, REF\_TICK is used.

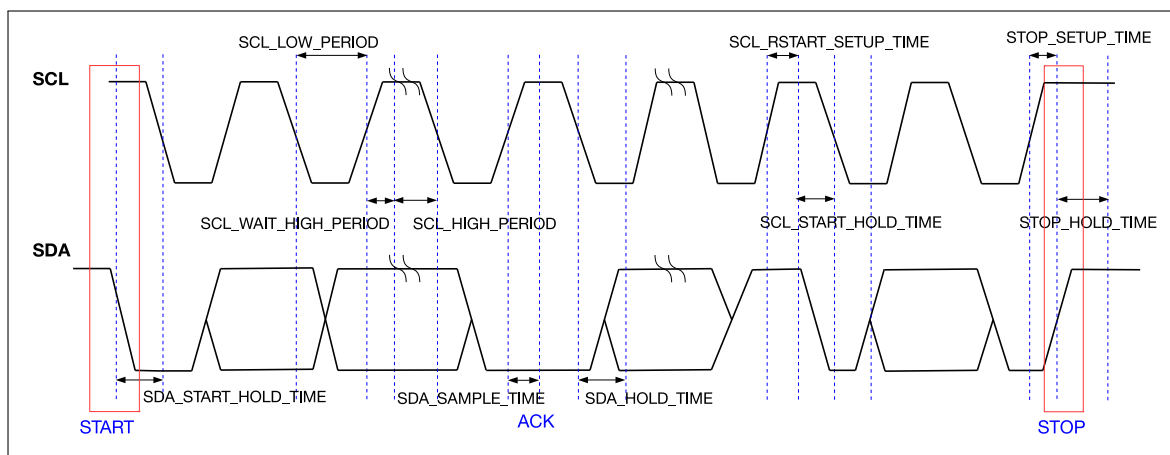


Figure 17-4. I2C Timing Diagram

Figure 17-4 shows the timing diagram of an I2C master. The unit of parameters in this figure is `I2C_CLK` ( $T_{I2C\_CLK}$ ). Specifically, when `I2C_REF_ALWAYS_ON` = 1,  $T_{I2C\_CLK}$  is  $T_{APB\_CLK}$ ; when

$I2C\_REF\_ALWAYS\_ON = 0$ ,  $T_{I2C\_CLK}$  is  $T_{REF\_TICK}$ . Figure 17-4 also specifies registers used to configure the START bit, STOP bit, data hold time, data sample time, waiting time on the rising SCL edge, etc. Parameters in Figure 17-4 are described as the following:

1.  **$I2C\_SCL\_START\_HOLD\_TIME$** : Specifies the interval between pulling SDA low and pulling SCL low when the master generates a START condition. This interval is  $(I2C\_SCL\_START\_HOLD\_TIME + 1) \times T_{I2C\_CLK}$ . This register is active only when the I2C controller works in master mode.
2.  **$I2C\_SCL\_LOW\_PERIOD$** : Specifies the low period of SCL. This period lasts  $(I2C\_SCL\_START\_HOLD\_TIME + 1) \times T_{I2C\_CLK}$ . However, it could be extended when SCL is pulled low by peripheral devices or by an END command executed by the I2C controller, or when the clock is stretched. This register is active only when the I2C controller works in master mode.
3.  **$I2C\_SCL\_WAIT\_HIGH\_PERIOD$** : Specifies time for SCL to go high in  $T_{I2C\_CLK}$ . Please make sure that SCL will be pulled high within this time period. Otherwise, the high period of SCL may be incorrect. This register is active only when the I2C controller works in master mode.
4.  **$I2C\_SCL\_HIGH\_PERIOD$** : Specifies the high period of SCL in  $T_{I2C\_CLK}$ . This register is active only when the I2C controller works in master mode. When SCL goes high within  $(I2C\_SCL\_WAIT\_HIGH\_PERIOD + 1) \times T_{I2C\_CLK}$ , its frequency is:

$$f_{scl} = \frac{f_{I2C\_CLK}}{I2C\_SCL\_LOW\_PERIOD + 1 + I2C\_SCL\_HIGH\_PERIOD + I2C\_SCL\_WAIT\_HIGH\_PERIOD}$$

5.  **$I2C\_SDA\_SAMPLE\_TIME$** : Specifies the interval between the rising edge of SCL and the level sampling time of SDA. It is advised to set a value in the middle of SCL's high period. This register is active both in master mode and slave mode.
6.  **$I2C\_SDA\_HOLD\_TIME$** : Specifies the interval between changing the SDA output level and the falling edge of SCL. This register is active both in master mode and slave mode.

SCL and SDA output drivers must be configured as open drain. There are two ways to achieve this:

1. Set  **$I2C\_SCL\_FORCE\_OUT$**  and  **$I2C\_SDA\_FORCE\_OUT$** , and configure  **$GPIO\_PINn\_PAD\_DRIVER$**  register for corresponding SCL and SDA pads as open-drain.
2. Clear  **$I2C\_SCL\_FORCE\_OUT$**  and  **$I2C\_SDA\_FORCE\_OUT$** .

Because these lines are configured as open-drain, the low to high transition time of each line is determined together by the pull-up resistance and the capacitance on the line. The output frequency of I2C is limited by the SDA and SCL line's pull-up speed, mainly SCL's speed.

In addition, when  **$I2C\_SCL\_FORCE\_OUT$**  and  **$I2C\_SCL\_PD\_EN$**  are set to 1, SCL can be forced low; when  **$I2C\_SDA\_FORCE\_OUT$**  and  **$I2C\_SDA\_PD\_EN$**  are set to 1, SDA can be forced low.

## 17.4 Typical Applications

For the convenience of description, I2C masters and slaves in all subsequent figures refer to ESP32-S2 I2C peripheral controllers. However, these controllers can communicate with any other I2C devices.

### 17.4.1 An I2C Master Writes to an I2C Slave with a 7-bit Address in One Command Sequence

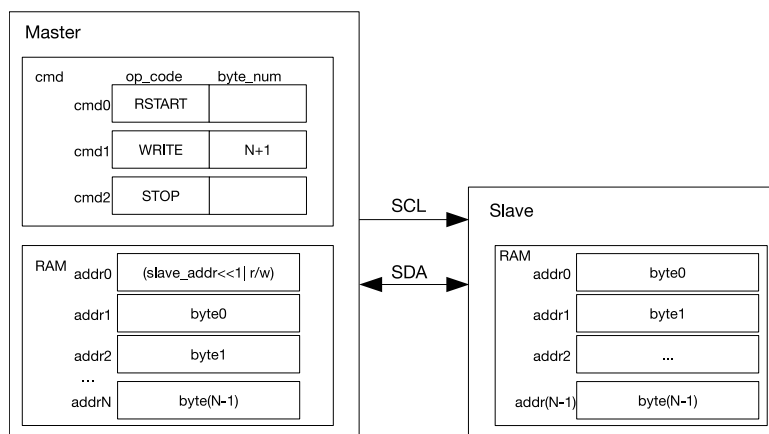


Figure 17-5. An I2C Master Writing to an I2C Slave with a 7-bit Address

Figure 17-5 shows how an I2C master writes N bytes of data using 7-bit addressing. As shown in figure 17-5, the first byte in the master's RAM is a 7-bit slave address followed by a  $R/\overline{W}$  bit. When the  $R/\overline{W}$  bit is zero, it indicates a WRITE operation. The remaining bytes are used to store data ready for transfer. The cmd box contains related command sequences.

After the command sequence is configured and data in RAM is ready, the master enables the controller and initiates data transfer by setting `I2C_TRANS_START` bit. The controller has four steps to take:

1. Wait for SCL to go high, to avoid SCL used by other masters or slaves.
2. Execute a RSTART command and send a START bit.
3. Execute a WRITE command by taking N+1 bytes from the RAM in order and send them to the slave in the same order. The first byte is the address of the slave.
4. Send a STOP. Once the I2C master transfers a STOP bit, an `I2C_TRANS_COMPLETE_INT` interrupt is generated.

If data to be transferred is larger than 32 bytes, the RAM access will wrap around. While the controller sends data, software must replace data already sent in RAM. To assist with this process, the master will generate an `I2C_TXFIFO_WM_INT` interrupt when less than `I2C_TXFIFO_WM_THRHD` bytes of data remain to be sent.

After detecting this interrupt, software can refresh data in RAM. When the RAM is accessed in non-FIFO mode, in order to overwrite existing data in the RAM with new ones, software needs to first configure `I2C_TX_UPDATE` bit to latch the start address and the end address of data sent in the RAM, and then read `I2C_TXFIFO_START_ADDR` and `I2C_TXFIFO_END_ADDR` field in `I2C_FIFO_ST_REG` register to obtain these addresses. When the RAM is accessed in FIFO mode, new data can be written to the RAM directly via `I2C_DATA_REG` register.

The controller stops executing the command sequence after a STOP command, or when one of the following two events occurs:

1. When `ack_check_en` is set to 1, the I2C master checks the ACK value each time it sends a data byte. If the ACK value received does not match `ack_exp` (the expected ACK value) in the WRITE command, the master generates an `I2C_NACK_INT` interrupt and stops the transmission.

- During the high period of SCL, if the input value and the output value of SDA do not match, the I2C master generates an I2C\_ARBITRATION\_LOST\_INT interrupt, stops executing the command sequence, returns to IDLE state and releases SCL and SDA.

Once detecting a START bit sent by the I2C master, the I2C slave receives the address and compares it with its own address. If the received address does not match I2C\_SLAVE\_ADDR[6:0], the slave stops receiving data. If the received address does match I2C\_SLAVE\_ADDR[6:0], the slave receives data and stores them into the RAM in order.

If data to be transferred is larger than 32 bytes, the RAM may wrap around. While the controller receives data, software reclaim data already received by the slave. To assist with this process, the master will generate an I2C\_RXFIFO\_WM\_INT after I2C\_RXFIFO\_WM\_THRHD bytes are received in RAM.

Once detecting this interrupt, software can read data from the RAM registers. When the RAM is accessed in non-FIFO mode, in order to read data, software needs to configure I2C\_RX\_UPDATE bit to latch the start address and the end address of data to be reclaimed, and read I2C\_RXFIFO\_START\_ADDR and I2C\_RXFIFO\_END\_ADDR fields in I2C\_FIFO\_ST\_REG register to obtain these addresses. When the RAM is accessed in FIFO mode, data can be read directly via I2C\_DATA\_REG register.

### 17.4.2 An I2C Master Writes to an I2C Slave with a 10-bit Address in One Command Sequence

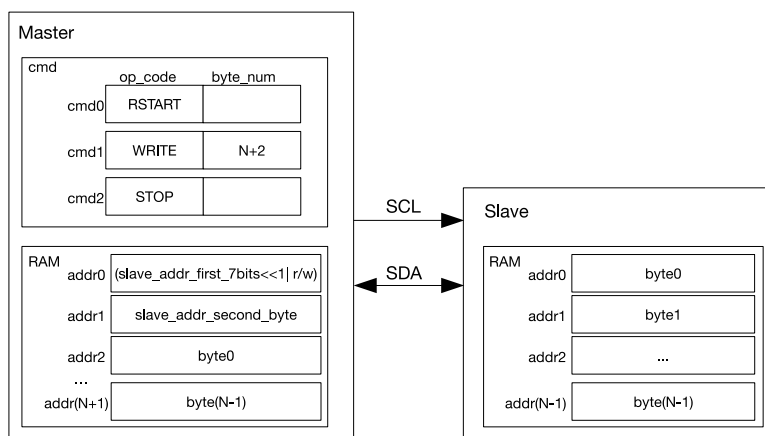


Figure 17-6. A Master Writing to a Slave with a 10-bit Address

Besides 7-bit addressing (SLV\_ADDR[6:0]), the ESP32-S2 I2C controller also supports 10-bit addressing (SLV\_ADDR[9:0]). In the following text, the slave address is referred to as SLV\_ADDR.

Figure 17-6 shows how an I2C master writes N-bytes of data using 10-bit addressing. Unlike a 7-bit address, a 10-bit slave address is formed from two bytes. In master mode, the first byte of the slave address, which comprises slave\_addr\_first\_7bits and a  $R/\overline{W}$  bit, is stored into addr0 in the RAM. slave\_addr\_first\_7bits should be configured as (0x78 | SLV\_ADDR[9:8]). The second byte slave\_addr\_second\_byte is stored into addr1 in the RAM. slave\_addr\_second\_byte should be configured as SLV\_ADDR[7:0].

In the slave, the 10-bit addressing mode is enabled by configuring I2C\_ADDR\_10BIT\_EN bit. The address of the I2C slave is configured using I2C\_SLAVE\_ADDR. I2C\_SLAVE\_ADDR[14:7] should be configured as SLV\_ADDR[7:0], and I2C\_SLAVE\_ADDR[6:0] should be configured as (0x78 | SLV\_ADDR[9:8]). Since a 10-bit slave address has one more byte than a 7-bit address, byte\_num of the WRITE command and the number of bytes in the RAM increase by one.

### 17.4.3 An I2C Master Writes to an I2C Slave with Two 7-bit Addresses in One Command Sequence

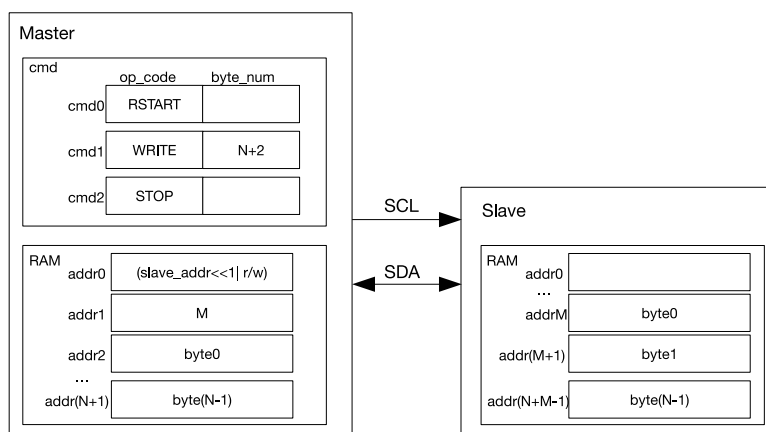


Figure 17-7. An I2C Master Writing Address M in the RAM to an I2C Slave with a 7-bit Address

When working in slave mode, the controller supports double addressing, where the first address is the address of an I2C slave, and the second one is the slave's memory address. Double addressing is enabled by configuring [I2C\\_FIFO\\_ADDR\\_CFG\\_EN](#). When using double addressing, RAM must be accessed in non-FIFO mode. As figure 17-7 illustrates, the I2C slave put received byte0 ~ byte(N-1) into its RAM in an order starting from **addrM**. The RAM is overwritten every 32 bytes.

### 17.4.4 An I2C Master Writes to an I2C Slave with a 7-bit Address in Multiple Command Sequences

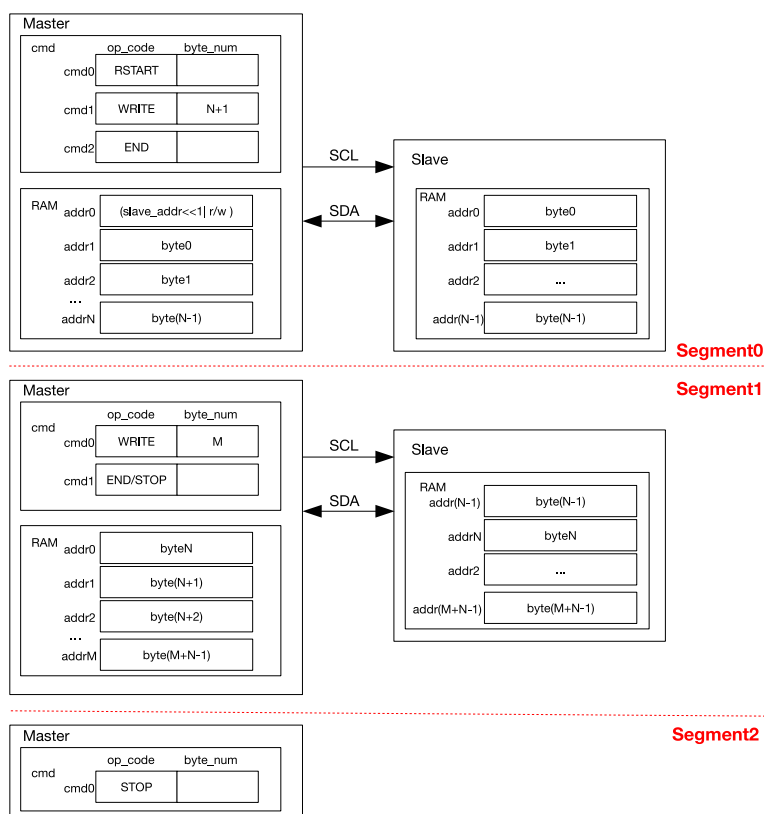


Figure 17-8. An I2C Master Writing to an I2C Slave with a 7-bit Address in Multiple Sequences

Given that the I2C Controller RAM holds only 32 bytes, when data are too large to be processed even by the wrapped RAM, it is advised to transmit them in multiple command sequences. At the end of every command sequence is an END command. When the controller executes this END command to pull SCL low, software refreshes command sequence registers and the RAM for next the transmission.

Figure 17-8 shows how an I2C master writes to an I2C slave in two or three segments. For the first segment, the CMD\_Controller registers are configured as shown in Segment0. Once data in the master's RAM is ready and `I2C_TRANS_START` is set, the master initiates data transfer. After executing the END command, the master turns off the SCL clock and pulls the SCL low to reserve the bus. Meanwhile, the controller generates an `I2C_END_DETECT_INT` interrupt.

For the second segment, after detecting the `I2C_END_DETECT_INT` interrupt, software refreshes the CMD\_Controller registers and reloads the RAM and clears this interrupt, as shown in Segment1. If cmd1 in the second segment is a STOP, then data is transmitted to the slave in two segments. The master resumes data transfer after `I2C_TRANS_START` is set, and terminates the transfer by sending a STOP bit.

For the third segment, after the second data transfer finishes and an `I2C_END_DETECT_INT` is detected, the CMD\_Controller registers of the master are configured as shown in Segment2. Once `I2C_TRANS_START` is set, the master generates a STOP bit and terminates the transfer.

Please note that other I2C masters will not transact on the bus between two segments. The bus is only released after a STOP signal is sent. To interrupt the transfer, the I2C controller can be reset by setting `I2C_FSM_RST` at any time. This register will later be cleared automatically by hardware.

When the master is in IDLE state and `I2C_SCL_RST_SLV_EN` is set, hardware sends `I2C_SCL_RST_SLV_NUM` SCL pulses. `I2C_SCL_RST_SLV_EN` will later be cleared automatically by hardware.

Note that the operation of other I2C masters and I2C slaves may differ from that of the ESP32-S2 I2C devices. Please refer to datasheets of specific I2C devices.

#### 17.4.5 An I2C Master Reads an I2C Slave with a 7-bit Address in One Command Sequence

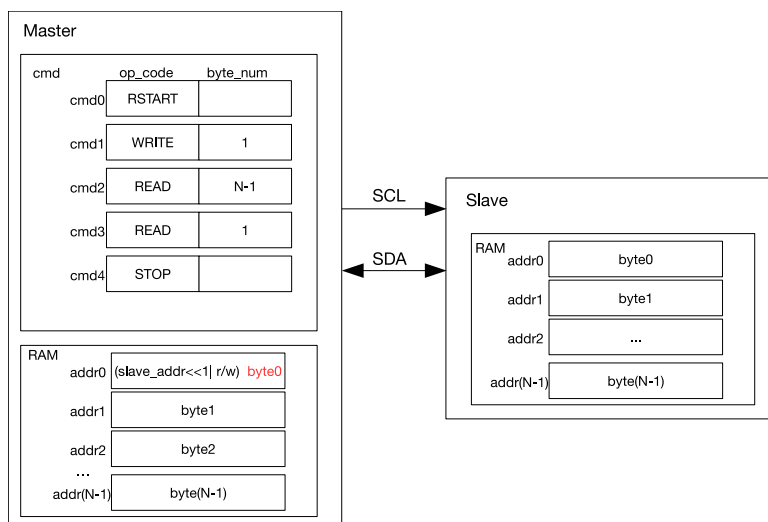


Figure 17-9. An I2C Master Reading an I2C Slave with a 7-bit Address

Figure 17-9 shows how an I2C master reads N bytes of data from an I2C slave using 7-bit addressing. cmd1 is a WRITE command, and when this command is executed the master sends a slave address. The byte sent



comprises a 7-bit slave address and a  $R/\overline{W}$  bit. When the  $R/\overline{W}$  bit is 1, it indicates a READ operation. If the address of an I2C slave matches the sent address, this matching slave starts sending data to the master. The master generates acknowledgements according to `ack_value` defined in the READ command upon receiving every byte.

As illustrated in Figure 17-9, the master executes two READ commands: it generates ACKs for (N-1) bytes of data in `cmd2`, and a NACK for the last byte of data in `cmd3`. This configuration may be changed as required. The master writes received data into the controller RAM from `addr0`, whose original content (a slave address and a  $R/\overline{W}$  bit) is overwritten by `byte0` marked red in Figure 17-9.

#### 17.4.6 An I2C Master Reads an I2C Slave with a 10-bit Address in One Command Sequence

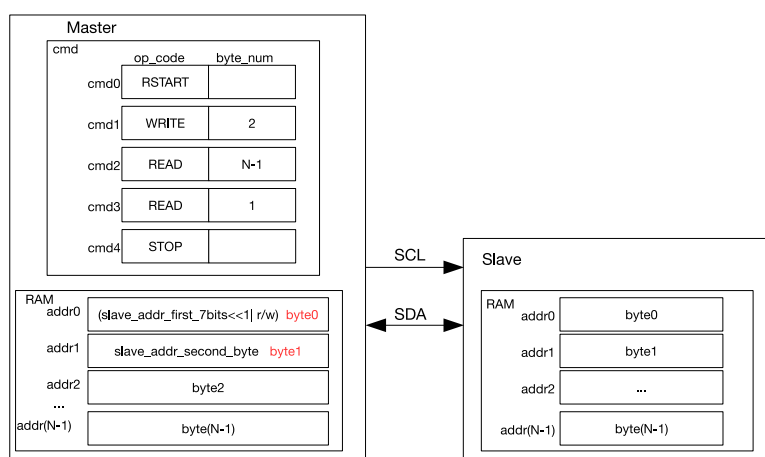
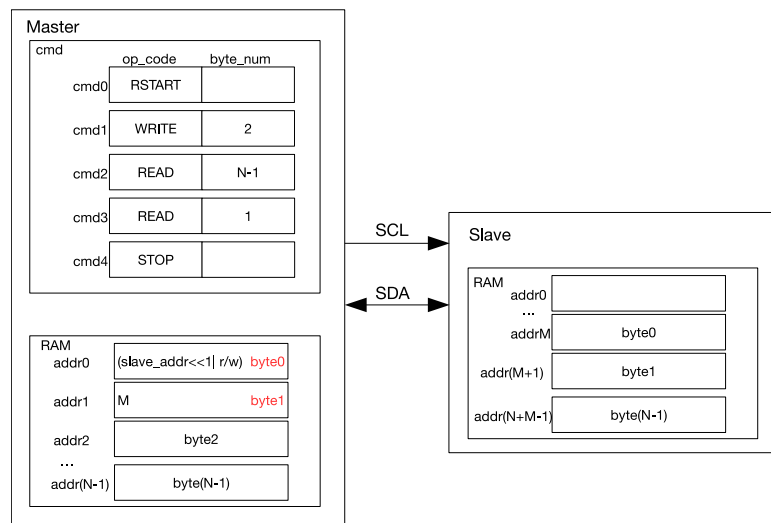


Figure 17-10. An I2C Master Reading an I2C Slave with a 10-bit Address

Figure 17-10 shows how an I2C master reads data from an I2C slave using 10-bit addressing. Unlike 7-bit addressing, in 10-bit addressing the WRITE command of the I2C master is formed from two bytes, and correspondingly the RAM of this master stores a 10-bit address of two bytes. `I2C_ADDR_10BIT_EN` and `I2C_SLAVE_ADDR[14:0]` should be set. Please refer to 17.4.2 for how to set this register.

### 17.4.7 An I2C Master Reads an I2C Slave with Two 7-bit Addresses in One Command Sequence



**Figure 17-11. An I2C Master Reading N Bytes of Data from addrM of an I2C Slave with a 7-bit Address**

Figure 17-11 shows how an I2C master reads data from specified addresses in an I2C slave. This procedure is as follows:

1. Set [I2C\\_FIFO\\_ADDR\\_CFG\\_EN](#) and prepare data to be sent in the RAM of the slave.
2. Prepare the slave address and register address M in the master.
3. Set [I2C\\_TRANS\\_START](#) and start transferring N bytes of data in the slave's RAM starting from address M to the master.

### 17.4.8 An I2C Master Reads an I2C Slave with a 7-bit Address in Multiple Command Sequences

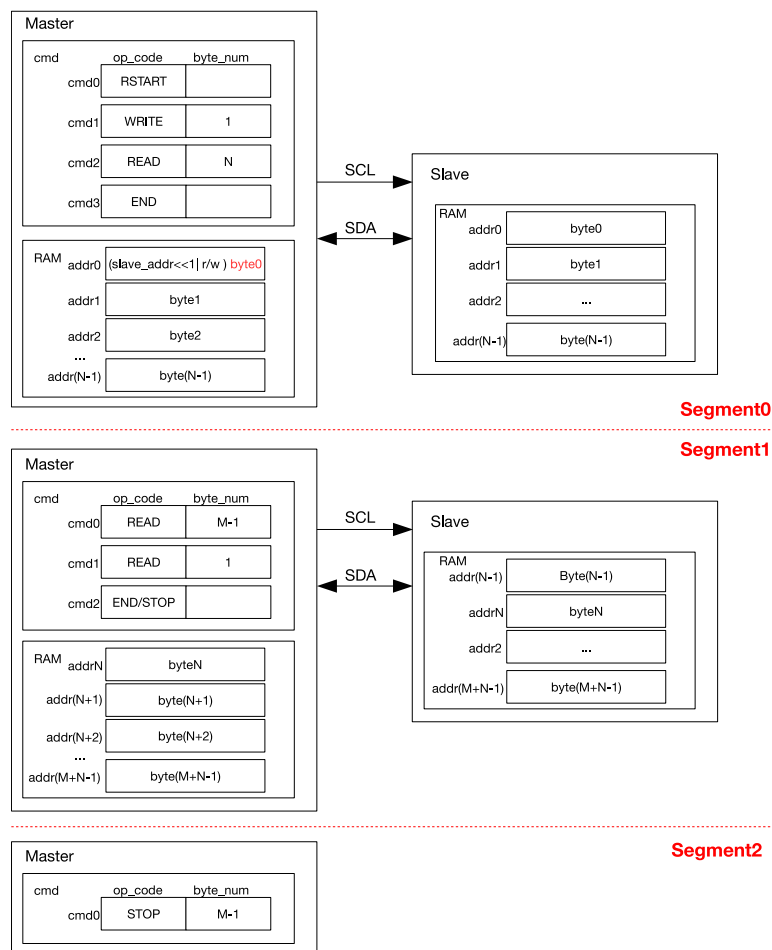


Figure 17-12. An I2C Master Reading an I2C Slave with a 7-bit Address in Segments

Figure 17-12 shows how an I2C master reads (N+M) bytes of data from an I2C slave in two/three segments separated by END commands. Configuration procedures are described as follows:

1. Configure the command register and the RAM, as shown in Segment0.
2. Prepare data in the RAM of the slave, and set `I2C_TRANS_START` to start data transfer. After executing the END command, the master refreshes command registers and the RAM as shown in Segment1, and clears the corresponding `I2C_END_DETECT_INT` interrupt. If cmd2 in the second segment is a STOP, then data is read from the slave in two segments. The master resumes data transfer by setting `I2C_TRANS_START` and terminates the transfer by sending a STOP bit.
3. If cmd2 in Segment1 is an END, then data is read from the slave in three segments. After the second data transfer finishes and an `I2C_END_DETECT_INT` interrupt is detected, the cmd box is configured as shown in Segment2. Once `I2C_TRANS_START` is set, the master terminates the transfer by sending a STOP bit.

## 17.5 Clock Stretching

In slave mode, the I2C controller can hold the SCL line low in exchange for more processing time. This function called clock stretching is enabled by setting `I2C_SLAVE_SCL_STRETCH_EN` bit. The time period of clock

stretching is configured by setting [I2C\\_STRETCH\\_PROTECT\\_NUM](#) bit. The SCL line will be held low when one of the following three events occurs:

1. Finding a match: In slave mode, the address of the I2C controller matches the address sent via the SDA line.
2. RAM being full: In slave mode, RX RAM of the I2C controller is full.
3. Data all sent: In slave mode, TX RAM of the I2C controller is empty.

After SCL has been stretched low, the cause of stretching can be read from [I2C\\_STRETCH\\_CAUSE](#) bit. Clock stretching is disabled by setting [I2C\\_SLAVE\\_SCL\\_STRETCH\\_CLR](#) bit.

## 17.6 Interrupts

- [I2C\\_SLAVE\\_STRETCH\\_INT](#): Generated when a slave holds SCL low.
- [I2C\\_DET\\_START\\_INT](#): Triggered when a START bit is detected.
- [I2C\\_SCL\\_MAIN\\_ST\\_TO\\_INT](#): Triggered when main state machine [SCL\\_MAIN\\_FSM](#) remains unchanged for over [I2C\\_SCL\\_MAIN\\_ST\\_TO](#)[23:0] clock cycles.
- [I2C\\_SCL\\_ST\\_TO\\_INT](#): Triggered when state machine [SCL\\_FSM](#) remains unchanged for over [I2C\\_SCL\\_ST\\_TO](#)[23:0] clock cycles.
- [I2C\\_RXFIFO\\_UDF\\_INT](#): Triggered when the I2C controller receives [I2C\\_NONFIFO\\_RX\\_THRES](#) bytes of data in non-FIFO mode.
- [I2C\\_TXFIFO\\_OVF\\_INT](#): Triggered when the I2C controller sends [I2C\\_NONFIFO\\_TX\\_THRES](#) bytes of data.
- [I2C\\_NACK\\_INT](#): Triggered when the ACK value received by the master is not as expected, or when the ACK value received by the slave is 1.
- [I2C\\_TRANS\\_START\\_INT](#): Triggered when the I2C controller sends a START bit.
- [I2C\\_TIME\\_OUT\\_INT](#): Triggered when SCL stays high or low for more than [I2C\\_TIME\\_OUT](#) clock cycles during data transfer.
- [I2C\\_TRANS\\_COMPLETE\\_INT](#): Triggered when the I2C controller detects a STOP bit.
- [I2C\\_MST\\_TXFIFO\\_UDF\\_INT](#): Triggered when TX FIFO of the master underflows.
- [I2C\\_ARBITRATION\\_LOST\\_INT](#): Triggered when the SDA's output value does not match its input value while the master's SCL is high.
- [I2C\\_BYTE\\_TRANS\\_DONE\\_INT](#): Triggered when the I2C controller sends or receives a byte.
- [I2C\\_END\\_DETECT\\_INT](#): Triggered when [op\\_code](#) of the master indicates an END command and an END condition is detected.
- [I2C\\_RXFIFO\\_OVF\\_INT](#): Triggered when Rx FIFO of the I2C controller overflows.
- [I2C\\_TXFIFO\\_WM\\_INT](#): I2C TX FIFO watermark interrupt. Triggered when [I2C\\_FIFO\\_PRT\\_EN](#) is 1 and the pointers of TX FIFO are less than [I2C\\_TXFIFO\\_WM\\_THRHD](#)[4:0].
- [I2C\\_RXFIFO\\_WM\\_INT](#): I2C Rx FIFO watermark interrupt. Triggered when [I2C\\_FIFO\\_PRT\\_EN](#) is 1 and the pointers of Rx FIFO are greater than [I2C\\_RXFIFO\\_WM\\_THRHD](#)[4:0].

## 17.7 Base Address

Users can access the I2C Controller with base addresses in Table 66. For more information about accessing peripherals from different buses please see Chapter 1 *System and Memory*.

**Table 66: I2C Controller Base Address**

Module	Bus to Access Peripheral	Base Address
I2C0	PeriBUS1	0x3F413000
	PeriBUS2	0x60013000
I2C1	PeriBUS1	0x3F427000
	PeriBUS2	0x60027000

## 17.8 Register Summary

Addresses in the following table are relative to I2C base addresses provided in Section 17.7.

Name	Description	Address	Access
<b>Timing Register</b>			
<a href="#">I2C_SCL_LOW_PERIOD_REG</a>	Configures the low level width of the SCL clock	0x0000	R/W
<a href="#">I2C_SDA_HOLD_REG</a>	Configures the hold time after a negative SCL edge	0x0030	R/W
<a href="#">I2C_SDA_SAMPLE_REG</a>	Configures the sample time after a positive SCL edge	0x0034	R/W
<a href="#">I2C_SCL_HIGH_PERIOD_REG</a>	Configures the high level width of the SCL clock	0x0038	R/W
<a href="#">I2C_SCL_START_HOLD_REG</a>	Configures the interval between pulling SDA low and pulling SCL low when the master generates a START condition	0x0040	R/W
<a href="#">I2C_SCL_RSTART_SETUP_REG</a>	Configures the interval between the positive edge of SCL and the negative edge of SDA	0x0044	R/W
<a href="#">I2C_SCL_STOP_HOLD_REG</a>	Configures the delay after the SCL clock edge for a stop condition	0x0048	R/W
<a href="#">I2C_SCL_STOP_SETUP_REG</a>	Configures the delay between the SDA and SCL positive edge for a stop condition	0x004C	R/W
<a href="#">I2C_SCL_ST_TIME_OUT_REG</a>	SCL status time out register	0x0098	R/W
<a href="#">I2C_SCL_MAIN_ST_TIME_OUT_REG</a>	SCL main status time out register	0x009C	R/W
<b>Configuration Register</b>			
<a href="#">I2C_CTR_REG</a>	Transmission setting	0x0004	R/W
<a href="#">I2C_TO_REG</a>	Setting time out control for receiving data	0x000C	R/W
<a href="#">I2C_SLAVE_ADDR_REG</a>	Local slave address setting	0x0010	R/W
<a href="#">I2C_FIFO_CONF_REG</a>	FIFO configuration register	0x0018	R/W
<a href="#">I2C_SCL_SP_CONF_REG</a>	Power configuration register	0x00A0	R/W
<a href="#">I2C_SCL_STRETCH_CONF_REG</a>	Set SCL stretch of I2C slave	0x00A4	varies
<b>Status Register</b>			
<a href="#">I2C_SR_REG</a>	Describe I2C work status	0x0008	RO
<a href="#">I2C_FIFO_ST_REG</a>	FIFO status register	0x0014	varies
<a href="#">I2C_DATA_REG</a>	RX FIFO read data	0x001C	RO

Name	Description	Address	Access
<b>Interrupt Register</b>			
<a href="#">I2C_INT_RAW_REG</a>	Raw interrupt status	0x0020	RO
<a href="#">I2C_INT_CLR_REG</a>	Interrupt clear bits	0x0024	WO
<a href="#">I2C_INT_ENA_REG</a>	Interrupt enable bits	0x0028	R/W
<a href="#">I2C_INT_STATUS_REG</a>	Status of captured I2C communication events	0x002C	RO
<b>Filter Register</b>			
<a href="#">I2C_SCL_FILTER_CFG_REG</a>	SCL filter configuration register	0x0050	R/W
<a href="#">I2C_SDA_FILTER_CFG_REG</a>	SDA filter configuration register	0x0054	R/W
<b>Command Register</b>			
<a href="#">I2C_COMD0_REG</a>	I2C command register 0	0x0058	R/W
<a href="#">I2C_COMD1_REG</a>	I2C command register 1	0x005C	R/W
<a href="#">I2C_COMD2_REG</a>	I2C command register 2	0x0060	R/W
<a href="#">I2C_COMD3_REG</a>	I2C command register 3	0x0064	R/W
<a href="#">I2C_COMD4_REG</a>	I2C command register 4	0x0068	R/W
<a href="#">I2C_COMD5_REG</a>	I2C command register 5	0x006C	R/W
<a href="#">I2C_COMD6_REG</a>	I2C command register 6	0x0070	R/W
<a href="#">I2C_COMD7_REG</a>	I2C command register 7	0x0074	R/W
<a href="#">I2C_COMD8_REG</a>	I2C command register 8	0x0078	R/W
<a href="#">I2C_COMD9_REG</a>	I2C command register 9	0x007C	R/W
<a href="#">I2C_COMD10_REG</a>	I2C command register 10	0x0080	R/W
<a href="#">I2C_COMD11_REG</a>	I2C command register 11	0x0084	R/W
<a href="#">I2C_COMD12_REG</a>	I2C command register 12	0x0088	R/W
<a href="#">I2C_COMD13_REG</a>	I2C command register 13	0x008C	R/W
<a href="#">I2C_COMD14_REG</a>	I2C command register 14	0x0090	R/W
<a href="#">I2C_COMD15_REG</a>	I2C command register 15	0x0094	R/W
<b>Version Register</b>			
<a href="#">I2C_DATE_REG</a>	Version control register	0x00F8	R/W

## 17.9 Registers

Register 17.1: I2C\_SCL\_LOW\_PERIOD\_REG (0x0000)

(reserved)																I2C_SCL_LOW_PERIOD																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																14																13																0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0</															

**I2C\_SCL\_LOW\_PERIOD** This register is used to configure for how long SCL remains low in master mode, in I2C module clock cycles. (R/W)

**Register 17.2: I2C\_SDA\_HOLD\_REG (0x0030)**

(reserved)																						I2C_SDA_HOLD_TIME																			
31																						9										0									
0 0																						0x0										Reset									

**I2C\_SDA\_HOLD\_TIME** This register is used to configure the interval between changing the SDA output level and the falling edge of SCL, in I2C module clock cycles. (R/W)

**Register 17.3: I2C\_SDA\_SAMPLE\_REG (0x0034)**

(reserved)																						I2C_SDA_SAMPLE_TIME																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																						10										9										0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
0																						0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0</									

**I2C\_SDA\_SAMPLE\_TIME** This register is used to configure the interval between the rising edge of SCL and the level sampling time of SDA, in I2C module clock cycles. (R/W)

**Register 17.4: I2C\_SCL\_HIGH\_PERIOD\_REG (0x0038)**

(reserved)				I2C_SCL_WAIT_HIGH_PERIOD										I2C_SCL_HIGH_PERIOD															
31				28				27				14				13				0									
0				0				0				0				0x00				0x00				Reset					

**I2C\_SCL\_HIGH\_PERIOD** This register is used to configure for how long SCL remains high in master mode, in I2C module clock cycles. (R/W)

**I2C\_SCL\_WAIT\_HIGH\_PERIOD** This register is used to configure for the SCL\_FSM's waiting period for SCL to go high in master mode, in I2C module clock cycles. (R/W)

**Register 17.5: I2C\_SCL\_START\_HOLD\_REG (0x0040)**

(reserved)																I2C_SCL_START_HOLD_TIME																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																10																9																0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0															

**I2C\_SCL\_START\_HOLD\_TIME** This register is used to configure interval between pulling SDA low and pulling SCL low when the master generates a START condition, in I2C module clock cycles. (R/W)

**Register 17.6: I2C\_SCL\_RSTART\_SETUP\_REG (0x0044)**

(reserved)																				I2C_SCL_RSTART_SETUP_TIME																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
31																				10										9										0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
0																				0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0										0									

**I2C\_SCL\_RSTART\_SETUP\_TIME** This register is used to configure the interval between the positive edge of SCL and the negative edge of SDA for a RESTART condition, in I2C module clock cycles. (R/W)

**Register 17.7: I2C\_SCL\_STOP\_HOLD\_REG (0x0048)**

(reserved)																I2C_SCL_STOP_HOLD_TIME													
31																0													
14																13													
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x00													
Reset																													

**I2C\_SCL\_STOP\_HOLD\_TIME** This register is used to configure the delay after the STOP condition, in I2C module clock cycles. (R/W)



**Register 17.8: I2C\_SCL\_STOP\_SETUP\_REG (0x004C)**

(reserved)																						I2C_SCL_STOP_SETUP_TIME																															
31																						10										9												0									
0 0																						0x0																						Reset									

**I2C\_SCL\_STOP\_SETUP\_TIME** This register is used to configure the time between the positive edge of SCL and the positive edge of SDA, in I2C module clock cycles. (R/W)

**Register 17.9: I2C\_SCL\_ST\_TIME\_OUT\_REG (0x0098)**

(reserved)								I2C_SCL_ST_TO																							
3124								230																							
00000000								0x0100Reset																							

**I2C\_SCL\_ST\_TO** The threshold value of SCL\_FSM state unchanged period. (R/W)

**Register 17.10: I2C\_SCL\_MAIN\_ST\_TIME\_OUT\_REG (0x009C)**

(reserved)								I2C_SCL_MAIN_ST_TO																																																															
31								24								23																							0																																
0								0								0								0								0								0								0								0x0100								Reset							

**I2C\_SCL\_MAIN\_ST\_TO** The threshold value of SCL\_MAIN\_FSM state unchanged period. (R/W)

## 394

ESP32-S2 TRM (Preliminary V0.5)

**I2C\_SCL\_FORCE\_OUT** 0: direct output; 1: open drain output. (R/W)

**I2C\_RX\_FULL\_ACK\_LEVEL** This register is used to configure the ACK value that need to sent by master when the rx\_fifo\_cnt has reached the threshold. (R/W)

**I2C\_MS\_MODE** Set this bit to configure the module as an I2C Master. Clear this bit to configure the module as an I2C Slave. (R/W)

**I2C\_TRANS\_START** Set this bit to start sending the data in TX FIFO. (R/W)

**I2C\_TX\_LSB\_FIRST** This bit is used to control the sending mode for data needing to be sent. 1: send data from the least significant bit; 0: send data from the most significant bit. (R/W)

**I2C\_RX\_LSB\_FIRST** This bit is used to control the storage mode for received data. 1: receive data from the least significant bit; 0: receive data from the most significant bit. (R/W)

**I2C\_CLK\_EN** Reserved (R/W)

**I2C\_ARBITRATION\_EN** This is the enable bit for I2C bus arbitration function. (R/W)

**I2C\_FSM\_RST** This register is used to reset the SCL\_FSM. (R/W)

**I2C\_REF\_ALWAYS\_ON** This register is used to control the REF\_TICK. (R/W)

### Register 17.12: I2C\_TO\_REG (0x000C)

(reserved)								I2C_TIME_OUT_EN																I2C_TIME_OUT_VALUE																
31								25								24	23																							0
0 0 0 0 0 0 0								0	0x0000																								Reset							

**I2C\_TIME\_OUT\_VALUE** This register is used to configure the timeout for receiving a data bit in APB clock cycles. (R/W)

**I2C\_TIME\_OUT\_EN** This is the enable bit for time out control. (R/W)

### Register 17.13: I2C\_SLAVE\_ADDR\_REG (0x0010)

The diagram illustrates the I2C Slave Address Register (I2C\_SLAVE\_ADDR) structure. It is a 32-bit register divided into three main sections:

- I2C\_ADDR\_10BIT\_EN**: A 2-bit field at the top left, labeled with a diagonal line.
- (reserved)**: A 2-bit field in the middle, labeled with a diagonal line.
- I2C\_SLAVE\_ADDR**: A 14-bit field at the bottom right, labeled with a diagonal line.

The register is shown with bit positions 31, 30, 15, 14, and 0. The I2C\_SLAVE\_ADDR field is currently set to 0x00.

**I2C\_SLAVE\_ADDR** When configured as an I2C Slave, this field is used to configure the slave address.  
(R/W)

**I2C\_ADDR\_10BIT\_EN** This field is used to enable the slave 10-bit addressing mode in master mode.  
(R/W)

## 396

ESP32-S2 TRM (Preliminary V0.5)

**I2C\_FIFO\_PRT\_EN** The control enable bit of FIFO pointer in non-FIFO mode. This bit controls the valid bits and the interrupts of TX/RX FIFO overflow, underflow, full and empty. (R/W)

**Register 17.15: I2C\_SCL\_SP\_CONF\_REG (0x00A0)**

(reserved)																								I2C_SDA_PD_EN		I2C_SCL_PD_EN		I2C_SCL_RST_SLV_NUM				I2C_SCL_RST_SLV_EN																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
31																								8	7	6	5	1				0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

**I2C\_SCL\_RST\_SLV\_EN** When I2C master is IDLE, set this bit to send out SCL pulses. The number of pulses equals to I2C\_SCL\_RST\_SLV\_NUM[4:0]. (R/W)

**I2C\_SCL\_RST\_SLV\_NUM** Configure the pulses of SCL generated in I2C master mode. Valid when I2C\_SCL\_RST\_SLV\_EN is 1. (R/W)

**I2C\_SCL\_PD\_EN** The power down enable bit for the I2C output SCL line. 1: Power down. 0: Not power down. Set I2C\_SCL\_FORCE\_OUT and I2C\_SCL\_PD\_EN to 1 to stretch SCL low. (R/W)

**I2C\_SDA\_PD\_EN** The power down enable bit for the I2C output SDA line. 1: Power down. 0: Not power down. Set I2C\_SDA\_FORCE\_OUT and I2C\_SDA\_PD\_EN to 1 to stretch SDA low. (R/W)

**Register 17.16: I2C\_SCL\_STRETCH\_CONF\_REG (0x00A4)**

(reserved)																				I2C_SLAVE_SCL_STRETCH_CLR		I2C_SLAVE_SCL_STRETCH_EN		I2C_STRETCH_PROTECT_NUM				
31												12		11	10	9	0											
0 0 0 0 0 0 0 0 0 0 0 0												0	0	0x0												Reset		

Reset

**I2C\_STRETCH\_PROTECT\_NUM** Configure the period of I2C slave stretching SCL line. (R/W)

**I2C\_SLAVE\_SCL\_STRETCH\_EN** The enable bit for slave SCL stretch function. 1: Enable. 0: Disable. The SCL output line will be stretched low when I2C\_SLAVE\_SCL\_STRETCH\_EN is 1 and stretch event happens. The stretch cause can be seen in I2C\_STRETCH\_CAUSE. (R/W)

**I2C\_SLAVE\_SCL\_STRETCH\_CLR** Set this bit to clear the I2C slave SCL stretch function. (WO)

**Register 17.17: I2C\_SR\_REG (0x0008)**

(reserved)		I2C_SCL_STATE_LAST		(reserved)		I2C_SCL_MAIN_STATE_LAST		I2C_TXFIFO_CNT		(reserved)		I2C_STRETCH_CAUSE		I2C_RXFIFO_CNT		(reserved)		I2C_BYTE_TRANS		I2C_SLAVE_ADDRESSED		I2C_BUS_BUSY		I2C_ARB_LOST		I2C_TIME_OUT		I2C_SLAVE_RW		I2C_RESP_REC	
31	30	28	27	26	24	23	18		17	16	15	14	13	8		7	6	5	4	3	2	1	0	Reset							
0	0x0	0	0x0	0x0		0		0	0x0	0x0		0		0	0	0	0	0	0	0	0	0	0								

**I2C\_RESP\_REC** The received ACK value in master mode or slave mode. 0: ACK; 1: NACK. (RO)

**I2C\_SLAVE\_RW** When in slave mode, 1: master reads from slave; 0: master writes to slave. (RO)

**I2C\_TIME\_OUT** When the I2C controller takes more than I2C\_TIME\_OUT clocks to receive a data bit, this field changes to 1. (RO)

**I2C\_ARB\_LOST** When the I2C controller loses control of SCL line, this register changes to 1. (RO)

**I2C\_BUS\_BUSY** 1: the I2C bus is busy transferring data; 0: the I2C bus is in idle state. (RO)

**I2C\_SLAVE\_ADDRESSED** When configured as an I2C Slave, and the address sent by the master is equal to the address of the slave, then this bit will be of high level. (RO)

**I2C\_BYTE\_TRANS** This field changes to 1 when one byte is transferred. (RO)

**I2C\_RXFIFO\_CNT** This field represents the amount of data needed to be sent. (RO)

**I2C\_STRETCH\_CAUSE** The cause of stretching SCL low in slave mode. 0: stretching SCL low at the beginning of I2C read data state. 1: stretching SCL low when I2C TX FIFO is empty in slave mode. 2: stretching SCL low when I2C RX FIFO is full in slave mode. (RO)

**I2C\_TXFIFO\_CNT** This field stores the amount of received data in RAM. (RO)

**I2C\_SCL\_MAIN\_STATE\_LAST** This field indicates the states of the I2C module state machine. 0: Idle; 1: Address shift; 2: ACK address; 3: RX data; 4: TX data; 5: Send ACK; 6: Wait ACK (RO)

**I2C\_SCL\_STATE\_LAST** This field indicates the states of the state machine used to produce SCL. 0: Idle; 1: Start; 2: Negative edge; 3: Low; 4: Positive edge; 5: High; 6: Stop (RO)

**Register 17.18: I2C\_FIFO\_ST\_REG (0x0014)**

(reserved)			I2C_SLAVE_RW_POINT				I2C_TX_UPDATE I2C_RX_UPDATE			I2C_TXFIFO_END_ADDR				I2C_TXFIFO_START_ADDR				I2C_RXFIFO_END_ADDR				I2C_RXFIFO_START_ADDR						
31	30	29	22				21	20	19	15				14	10				9	5				4	0			
0	0	0x0				0	0	0x0				0x0				0x0				0x0				0x0				Reset

**I2C\_RXFIFO\_START\_ADDR** This is the offset address of the last received data, as described in I2C\_NONFIFO\_RX\_THRES. (RO)

**I2C\_RXFIFO\_END\_ADDR** This is the offset address of the last received data, as described in I2C\_NONFIFO\_RX\_THRES. This value refreshes when an I2C\_RXFIFO\_UDF\_INT or I2C\_TRANS\_COMPLETE\_INT interrupt is generated. (RO)

**I2C\_TXFIFO\_START\_ADDR** This is the offset address of the first sent data, as described in I2C\_NONFIFO\_TX\_THRES. (RO)

**I2C\_TXFIFO\_END\_ADDR** This is the offset address of the last sent data, as described in I2C\_NONFIFO\_TX\_THRES. The value refreshes when an I2C\_TXFIFO\_OVF\_INT or I2C\_TRANS\_COMPLETE\_INT interrupt is generated. (RO)

**I2C\_RX\_UPDATE** Write 0 or 1 to I2C\_RX\_UPDATE to update the value of I2C\_RXFIFO\_END\_ADDR and I2C\_RXFIFO\_START\_ADDR. (WO)

**I2C\_TX\_UPDATE** Write 0 or 1 to I2C\_TX\_UPDATE to update the value of I2C\_TXFIFO\_END\_ADDR and I2C\_TXFIFO\_START\_ADDR. (WO)

**I2C\_SLAVE\_RW\_POINT** The received data in I2C slave mode. (RO)

**Register 17.19: I2C\_DATA\_REG (0x001C)**

(reserved)																I2C_FIFO_RDATA							
31																	8	7					0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0				Reset

**I2C\_FIFO\_RDATA** The value of RX FIFO read data. (RO)

Register 17.20: I2C\_INT\_RAW\_REG (0x0020)

(reserved)																																I2C_SLAVE_STRETCH_INT_RAW	I2C_DET_START_INT_RAW	I2C_SCL_MAIN_ST_TO_INT_RAW	I2C_SCL_ST_TO_INT_RAW	I2C_RXFIFO_UDF_INT_RAW	I2C_TXFIFO_OVF_INT_RAW	I2C_NACK_INT_RAW	I2C_TRANS_START_INT_RAW	I2C_TIME_OUT_INT_RAW	I2C_TRANS_COMPLETE_INT_RAW	I2C_MST_TXFIFO_UDF_INT_RAW	I2C_ARBITRATION_LOST_INT_RAW	I2C_BYTE_TRANS_DONE_INT_RAW	I2C_END_DETECT_INT_RAW	I2C_RXFIFO_WM_INT_RAW	I2C_TXFIFO_WM_INT_RAW	
31																17																16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset				

**I2C\_RXFIFO\_WM\_INT\_RAW** The raw interrupt bit for I2C\_RXFIFO\_WM\_INT interrupt. (RO)

**I2C\_TXFIFO\_WM\_INT\_RAW** The raw interrupt bit for I2C\_TXFIFO\_WM\_INT interrupt. (RO)

**I2C\_RXFIFO\_OVF\_INT\_RAW** The raw interrupt bit for I2C\_RXFIFO\_OVF\_INT interrupt. (RO)

**I2C\_END\_DETECT\_INT\_RAW** The raw interrupt bit for the I2C\_END\_DETECT\_INT interrupt. (RO)

**I2C\_BYTE\_TRANS\_DONE\_INT\_RAW** The raw interrupt bit for the I2C\_END\_DETECT\_INT interrupt. (RO)

**I2C\_ARBITRATION\_LOST\_INT\_RAW** The raw interrupt bit for the I2C\_ARBITRATION\_LOST\_INT interrupt. (RO)

**I2C\_MST\_TXFIFO\_UDF\_INT\_RAW** The raw interrupt bit for I2C\_TRANS\_COMPLETE\_INT interrupt. (RO)

**I2C\_TRANS\_COMPLETE\_INT\_RAW** The raw interrupt bit for the I2C\_TRANS\_COMPLETE\_INT interrupt. (RO)

**I2C\_TIME\_OUT\_INT\_RAW** The raw interrupt bit for the I2C\_TIME\_OUT\_INT interrupt. (RO)

**I2C\_TRANS\_START\_INT\_RAW** The raw interrupt bit for the I2C\_TRANS\_START\_INT interrupt. (RO)

**I2C\_NACK\_INT\_RAW** The raw interrupt bit for I2C\_SLAVE\_STRETCH\_INT interrupt. (RO)

**I2C\_TXFIFO\_OVF\_INT\_RAW** The raw interrupt bit for I2C\_TXFIFO\_OVF\_INT interrupt. (RO)

**I2C\_RXFIFO\_UDF\_INT\_RAW** The raw interrupt bit for I2C\_RXFIFO\_UDF\_INT interrupt. (RO)

**I2C\_SCL\_ST\_TO\_INT\_RAW** The raw interrupt bit for I2C\_SCL\_ST\_TO\_INT interrupt. (RO)

**I2C\_SCL\_MAIN\_ST\_TO\_INT\_RAW** The raw interrupt bit for I2C\_SCL\_MAIN\_ST\_TO\_INT interrupt. (RO)

**I2C\_DET\_START\_INT\_RAW** The raw interrupt bit for I2C\_DET\_START\_INT interrupt. (RO)

**I2C\_SLAVE\_STRETCH\_INT\_RAW** The raw interrupt bit for I2C\_SLAVE\_STRETCH\_INT interrupt. (RO)



Register 17.21: I2C\_INT\_CLR\_REG (0x0024)

(reserved)																I2C_SLAVE_STRETCH_INT_CLR I2C_DET_START_INT_CLR I2C_SCL_MAIN_ST_TO_INT_CLR I2C_SCL_ST_TO_INT_CLR I2C_RXFIFO_UDF_INT_CLR I2C_TXFIFO_UDF_INT_CLR I2C_NACK_INT_CLR I2C_TRANS_START_INT_CLR I2C_TIME_OUT_INT_CLR I2C_TRANS_COMPLETE_INT_CLR I2C_MST_TXFIFO_UDF_INT_CLR I2C_ARBTRATION_LOST_INT_CLR I2C_BYTE_TRANS_DONE_INT_CLR I2C_END_DETECT_INT_CLR I2C_RXFIFO_OVF_INT_CLR I2C_TXFIFO_WM_INT_CLR I2C_RXFIFO_WM_INT_CLR																		
31																17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**I2C\_RXFIFO\_WM\_INT\_CLR** Set this bit to clear I2C\_RXFIFO\_WM\_INT interrupt. (WO)

**I2C\_TXFIFO\_WM\_INT\_CLR** Set this bit to clear I2C\_TXFIFO\_WM\_INT interrupt. (WO)

**I2C\_RXFIFO\_OVF\_INT\_CLR** Set this bit to clear I2C\_RXFIFO\_OVF\_INT interrupt. (WO)

**I2C\_END\_DETECT\_INT\_CLR** Set this bit to clear the I2C\_END\_DETECT\_INT interrupt. (WO)

**I2C\_BYTE\_TRANS\_DONE\_INT\_CLR** Set this bit to clear the I2C\_END\_DETECT\_INT interrupt. (WO)

**I2C\_ARBTRATION\_LOST\_INT\_CLR** Set this bit to clear the I2C\_ARBTRATION\_LOST\_INT interrupt. (WO)

**I2C\_MST\_TXFIFO\_UDF\_INT\_CLR** Set this bit to clear I2C\_TRANS\_COMPLETE\_INT interrupt. (WO)

**I2C\_TRANS\_COMPLETE\_INT\_CLR** Set this bit to clear the I2C\_TRANS\_COMPLETE\_INT interrupt. (WO)

**I2C\_TIME\_OUT\_INT\_CLR** Set this bit to clear the I2C\_TIME\_OUT\_INT interrupt. (WO)

**I2C\_TRANS\_START\_INT\_CLR** Set this bit to clear the I2C\_TRANS\_START\_INT interrupt. (WO)

**I2C\_NACK\_INT\_CLR** Set this bit to clear I2C\_SLAVE\_STRETCH\_INT interrupt. (WO)

**I2C\_TXFIFO\_OVF\_INT\_CLR** Set this bit to clear I2C\_TXFIFO\_OVF\_INT interrupt. (WO)

**I2C\_RXFIFO\_UDF\_INT\_CLR** Set this bit to clear I2C\_RXFIFO\_UDF\_INT interrupt. (WO)

**I2C\_SCL\_ST\_TO\_INT\_CLR** Set this bit to clear I2C\_SCL\_ST\_TO\_INT interrupt. (WO)

**I2C\_SCL\_MAIN\_ST\_TO\_INT\_CLR** Set this bit to clear I2C\_SCL\_MAIN\_ST\_TO\_INT interrupt. (WO)

**I2C\_DET\_START\_INT\_CLR** Set this bit to clear I2C\_DET\_START\_INT interrupt. (WO)

**I2C\_SLAVE\_STRETCH\_INT\_CLR** Set this bit to clear I2C\_SLAVE\_STRETCH\_INT interrupt. (WO)

Register 17.22: I2C\_INT\_ENA\_REG (0x0028)

(reserved)																I2C_SLAVE_STRETCH_INT_ENA I2C_DET_START_INT_ENA I2C_SCL_MAIN_ST_TO_INT_ENA I2C_SCL_ST_TO_INT_ENA I2C_RXFIFO_UDF_INT_ENA I2C_TXFIFO_OVF_INT_ENA I2C_NACK_INT_ENA I2C_TIME_OUT_INT_ENA I2C_TRANS_START_INT_ENA I2C_TRANS_COMPLETE_INT_ENA I2C_ARBITRATION_LOST_INT_ENA I2C_BYTE_TRANS_DONE_INT_ENA I2C_END_DETECT_INT_ENA I2C_RXFIFO_OVF_INT_ENA I2C_TXFIFO_WM_INT_ENA I2C_RXFIFO_WM_INT_ENA																			
31																17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**I2C\_RXFIFO\_WM\_INT\_ENA** The raw interrupt bit for I2C\_RXFIFO\_WM\_INT interrupt. (R/W)

**I2C\_TXFIFO\_WM\_INT\_ENA** The raw interrupt bit for I2C\_TXFIFO\_WM\_INT interrupt. (R/W)

**I2C\_RXFIFO\_OVF\_INT\_ENA** The raw interrupt bit for I2C\_RXFIFO\_OVF\_INT interrupt. (R/W)

**I2C\_END\_DETECT\_INT\_ENA** The raw interrupt bit for the I2C\_END\_DETECT\_INT interrupt. (R/W)

**I2C\_BYTE\_TRANS\_DONE\_INT\_ENA** The raw interrupt bit for the I2C\_END\_DETECT\_INT interrupt. (R/W)

**I2C\_ARBITRATION\_LOST\_INT\_ENA** The raw interrupt bit for the I2C\_ARBITRATION\_LOST\_INT interrupt. (R/W)

**I2C\_MST\_TXFIFO\_UDF\_INT\_ENA** The raw interrupt bit for I2C\_TRANS\_COMPLETE\_INT interrupt. (R/W)

**I2C\_TRANS\_COMPLETE\_INT\_ENA** The raw interrupt bit for the I2C\_TRANS\_COMPLETE\_INT interrupt. (R/W)

**I2C\_TIME\_OUT\_INT\_ENA** The raw interrupt bit for the I2C\_TIME\_OUT\_INT interrupt. (R/W)

**I2C\_TRANS\_START\_INT\_ENA** The raw interrupt bit for the I2C\_TRANS\_START\_INT interrupt. (R/W)

**I2C\_NACK\_INT\_ENA** The raw interrupt bit for I2C\_SLAVE\_STRETCH\_INT interrupt. (R/W)

**I2C\_TXFIFO\_OVF\_INT\_ENA** The raw interrupt bit for I2C\_TXFIFO\_OVF\_INT interrupt. (R/W)

**I2C\_RXFIFO\_UDF\_INT\_ENA** The raw interrupt bit for I2C\_RXFIFO\_UDF\_INT interrupt. (R/W)

**I2C\_SCL\_ST\_TO\_INT\_ENA** The raw interrupt bit for I2C\_SCL\_ST\_TO\_INT interrupt. (R/W)

**I2C\_SCL\_MAIN\_ST\_TO\_INT\_ENA** The raw interrupt bit for I2C\_SCL\_MAIN\_ST\_TO\_INT interrupt. (R/W)

**I2C\_DET\_START\_INT\_ENA** The raw interrupt bit for I2C\_DET\_START\_INT interrupt. (R/W)

**I2C\_SLAVE\_STRETCH\_INT\_ENA** The raw interrupt bit for I2C\_SLAVE\_STRETCH\_INT interrupt. (R/W)

Register 17.23: I2C\_INT\_STATUS\_REG (0x002C)

(reserved)																I2C_SLAVE_STRETCH_INT_ST I2C_DET_START_INT_ST I2C_SCL_MAIN_ST_TO_INT_ST I2C_SCL_ST_TO_INT_ST I2C_RXFIFO_UDF_INT_ST I2C_TXFIFO_OVF_INT_ST I2C_NACK_INT_ST I2C_TRANS_START_INT_ST I2C_TIME_OUT_INT_ST I2C_TRANS_COMPLETE_INT_ST I2C_MST_TXFIFO_UDF_INT_ST I2C_ARBITRATION_LOST_INT_ST I2C_BYTE_TRANS_DONE_INT_ST I2C_END_DETECT_INT_ST I2C_RXFIFO_OVF_INT_ST I2C_RXFIFO_WM_INT_ST																		
31																	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset	

**I2C\_RXFIFO\_WM\_INT\_ST** The masked interrupt status bit for I2C\_RXFIFO\_WM\_INT interrupt. (RO)

**I2C\_TXFIFO\_WM\_INT\_ST** The masked interrupt status bit for I2C\_TXFIFO\_WM\_INT interrupt. (RO)

**I2C\_RXFIFO\_OVF\_INT\_ST** The masked interrupt status bit for I2C\_RXFIFO\_OVF\_INT interrupt. (RO)

**I2C\_END\_DETECT\_INT\_ST** The masked interrupt status bit for the I2C\_END\_DETECT\_INT interrupt. (RO)

**I2C\_BYTE\_TRANS\_DONE\_INT\_ST** The masked interrupt status bit for the I2C\_BYTE\_TRANS\_DONE\_INT interrupt. (RO)

**I2C\_ARBITRATION\_LOST\_INT\_ST** The masked interrupt status bit for the I2C\_ARBITRATION\_LOST\_INT interrupt. (RO)

**I2C\_MST\_TXFIFO\_UDF\_INT\_ST** The masked interrupt status bit for I2C\_MST\_TXFIFO\_UDF\_INT interrupt. (RO)

**I2C\_TRANS\_COMPLETE\_INT\_ST** The masked interrupt status bit for the I2C\_TRANS\_COMPLETE\_INT interrupt. (RO)

**I2C\_TIME\_OUT\_INT\_ST** The masked interrupt status bit for the I2C\_TIME\_OUT\_INT interrupt. (RO)

**I2C\_TRANS\_START\_INT\_ST** The masked interrupt status bit for the I2C\_TRANS\_START\_INT interrupt. (RO)

**I2C\_NACK\_INT\_ST** The masked interrupt status bit for I2C\_NACK\_INT interrupt. (RO)

**I2C\_TXFIFO\_OVF\_INT\_ST** The masked interrupt status bit for I2C\_TXFIFO\_OVF\_INT interrupt. (RO)

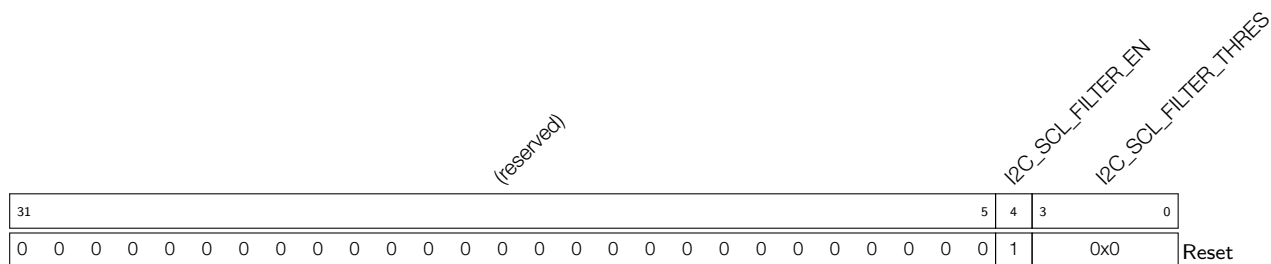
**I2C\_RXFIFO\_UDF\_INT\_ST** The masked interrupt status bit for I2C\_RXFIFO\_UDF\_INT interrupt. (RO)

**I2C\_SCL\_ST\_TO\_INT\_ST** The masked interrupt status bit for I2C\_SCL\_ST\_TO\_INT interrupt. (RO)

**I2C\_SCL\_MAIN\_ST\_TO\_INT\_ST** The masked interrupt status bit for I2C\_SCL\_MAIN\_ST\_TO\_INT interrupt. (RO)

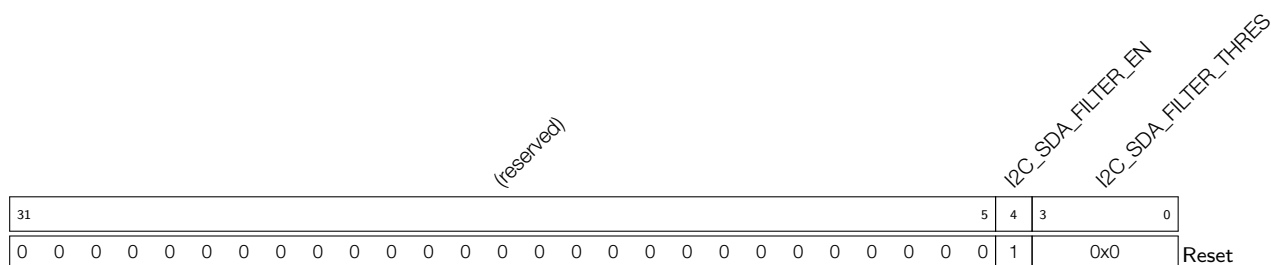
**I2C\_DET\_START\_INT\_ST** The masked interrupt status bit for I2C\_DET\_START\_INT interrupt. (RO)

**I2C\_SLAVE\_STRETCH\_INT\_ST** The masked interrupt status bit for I2C\_SLAVE\_STRETCH\_INT interrupt. (RO)

**Register 17.24: I2C\_SCL\_FILTER\_CFG\_REG (0x0050)**

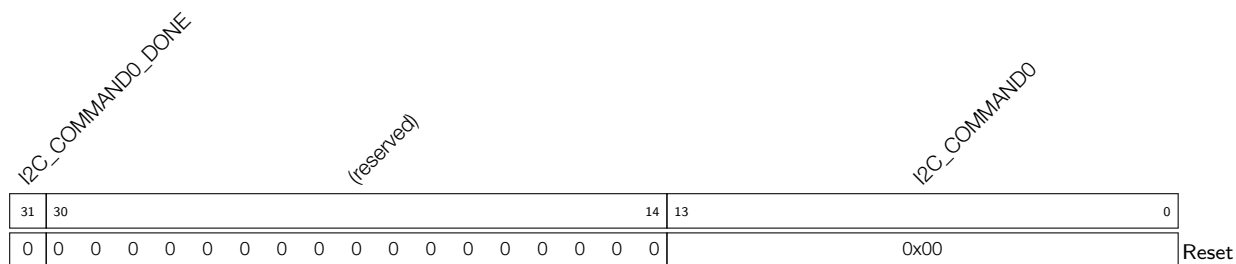
**I2C\_SCL\_FILTER\_THRES** When a pulse on the SCL input has smaller width than this register value in I2C module clock cycles, the I2C controller will ignore that pulse. (R/W)

**I2C\_SCL\_FILTER\_EN** This is the filter enable bit for SCL. (R/W)

**Register 17.25: I2C\_SDA\_FILTER\_CFG\_REG (0x0054)**

**I2C\_SDA\_FILTER\_THRES** When a pulse on the SDA input has smaller width than this register value in I2C module clock cycles, the I2C controller will ignore that pulse. (R/W)

**I2C\_SDA\_FILTER\_EN** This is the filter enable bit for SDA. (R/W)

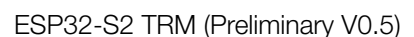
**Register 17.26: I2C\_CMD0\_REG (0x0058)**

**I2C\_COMMAND0** This is the content of command 0. It consists of three parts: op\_code is the command, 0: RSTART; 1: WRITE; 2: READ; 3: STOP; 4: END. byte\_num represents the number of bytes that need to be sent or received. ack\_check\_en, ack\_exp and ack are used to control the ACK bit. See I2C cmd structure for more Information. (R/W)

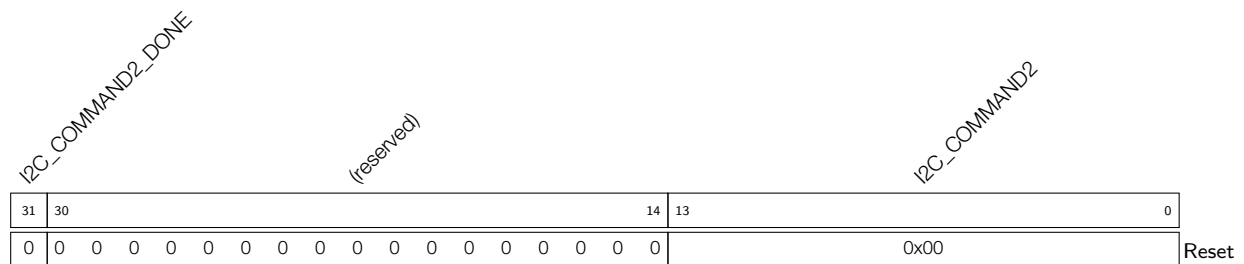
**I2C\_COMMAND0\_DONE** When command 0 is done in I2C Master mode, this bit changes to high level. (R/W)

405

[Submit Documentation Feedback](#)

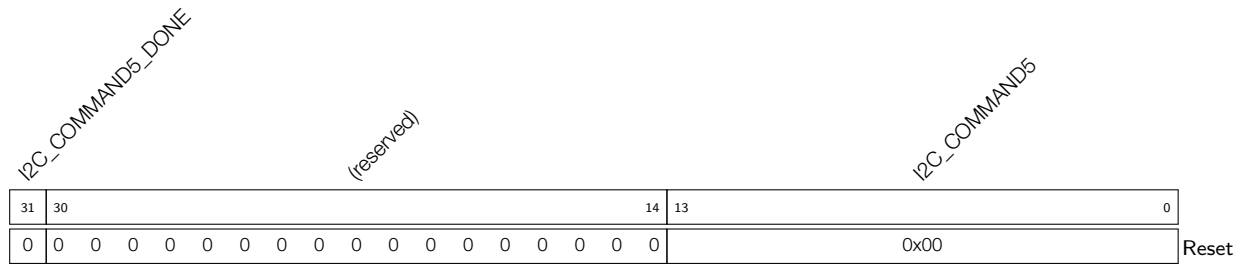


**I2C\_COMMAND1\_DONE** When command 1 is done in I2C Master mode, this bit changes to high level. (R/W)



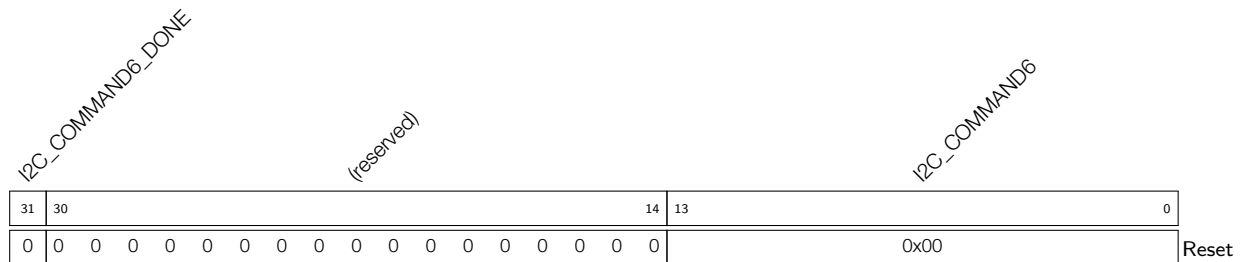
**I2C\_COMMAND2\_DONE** When command 2 is done in I2C Master mode, this bit changes to high Level. (R/W)



**Register 17.31: I2C\_COMD5\_REG (0x006C)**

**I2C\_COMMAND5** This is the content of command 5. It consists of three parts: op\_code is the command, 0: RSTART; 1: WRITE; 2: READ; 3: STOP; 4: END. byte\_num represents the number of bytes that need to be sent or received. ack\_check\_en, ack\_exp and ack are used to control the ACK bit. See I2C cmd structure for more Information. (R/W)

**I2C\_COMMAND5\_DONE** When command 5 is done in I2C Master mode, this bit changes to high level. (R/W)

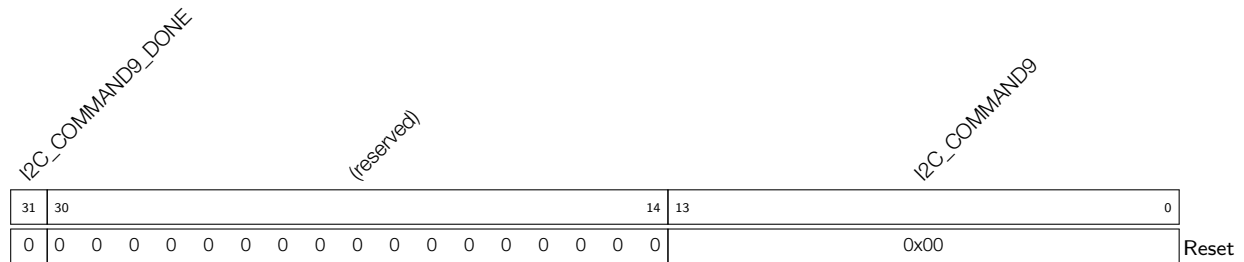
**Register 17.32: I2C\_COMD6\_REG (0x0070)**

**I2C\_COMMAND6** This is the content of command 6. It consists of three parts: op\_code is the command, 0: RSTART; 1: WRITE; 2: READ; 3: STOP; 4: END. byte\_num represents the number of bytes that need to be sent or received. ack\_check\_en, ack\_exp and ack are used to control the ACK bit. See I2C cmd structure for more Information. (R/W)

**I2C\_COMMAND6\_DONE** When command 6 is done in I2C Master mode, this bit changes to high level. (R/W)

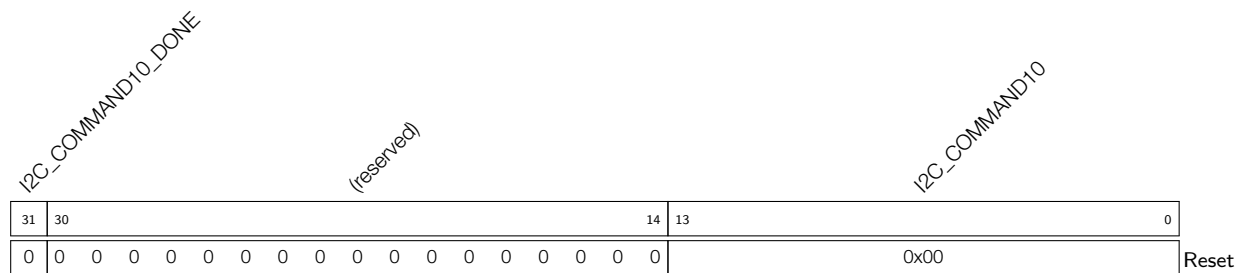




**Register 17.35: I2C\_COMD9\_REG (0x007C)**

**I2C\_COMMAND9** This is the content of command 9. It consists of three parts: op\_code is the command, 0: RSTART; 1: WRITE; 2: READ; 3: STOP; 4: END. byte\_num represents the number of bytes that need to be sent or received. ack\_check\_en, ack\_exp and ack are used to control the ACK bit. See I2C cmd structure for more Information. (R/W)

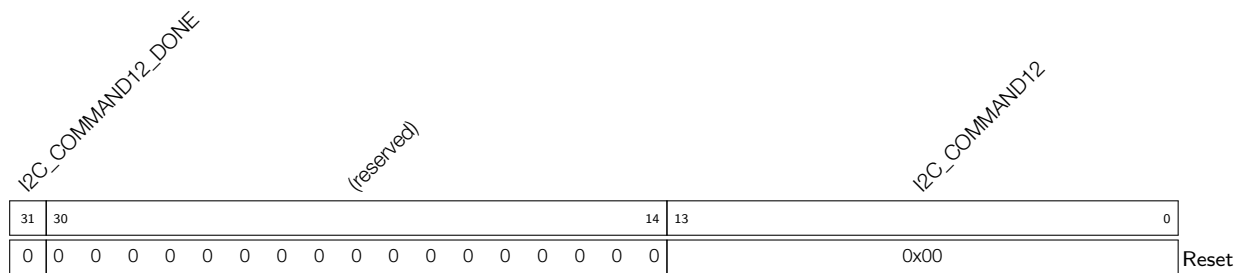
**I2C\_COMMAND9\_DONE** When command 9 is done in I2C Master mode, this bit changes to high level. (R/W)

**Register 17.36: I2C\_COMD10\_REG (0x0080)**

**I2C\_COMMAND10** This is the content of command 10. It consists of three parts: op\_code is the command, 0: RSTART; 1: WRITE; 2: READ; 3: STOP; 4: END. byte\_num represents the number of bytes that need to be sent or received. ack\_check\_en, ack\_exp and ack are used to control the ACK bit. See I2C cmd structure for more Information. (R/W)

**I2C\_COMMAND10\_DONE** When command 10 is done in I2C Master mode, this bit changes to high level. (R/W)

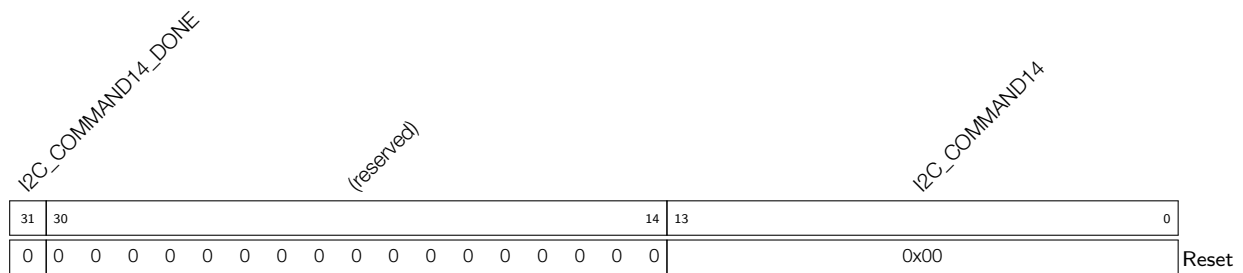
### Register 17.38: I2C\_COMMD12\_REG (0x0088)



**I2C\_COMMAND12** This is the content of command 12. It consists of three parts: op\_code is the command, 0: RSTART; 1: WRITE; 2: READ; 3: STOP; 4: END. byte\_num represents the number of bytes that need to be sent or received. ack\_check\_en, ack\_exp and ack are used to control the ACK bit. See I2C cmd structure for more Information. (R/W)

**I2C\_COMMAND12\_DONE** When command 12 is done in I2C Master mode, this bit changes to high level. (R/W)

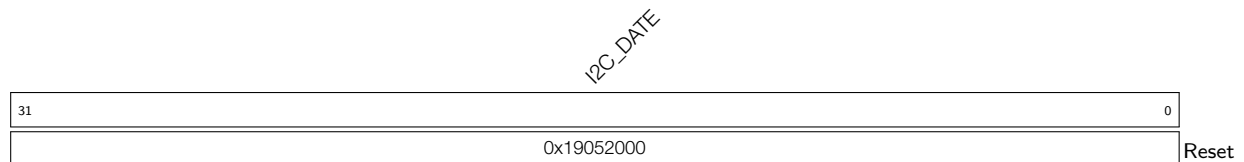
### Register 17.40: I2C\_COMMD14\_REG (0x0090)



**I2C\_COMMAND13\_DONE** When command 13 is done in I2C Master mode, this bit changes to high level. (R/W)

**I2C\_COMMAND14\_DONE** When command 14 is done in I2C Master mode, this bit changes to high level. (R/W)

### Register 17.42: I2C\_DATE\_REG (0x00F8)



**I2C\_DATE** This is the the version control register. (R/W)

## 18. TWAI

### 18.1 Overview

The Two-Wire Automobile Interface (TWAI) is a multi-master, multi-cast communication protocol with error detection and signaling and inbuilt message priorities and arbitration. The TWAI protocol is suited for automotive and industrial applications (Please see [TWAI Protocol Description](#)).

ESP32-S2 contains a TWAI controller that can be connected to a TWAI bus via an external transceiver. The TWAI controller contains numerous advanced features, and can be utilized in a wide range of use cases such as automotive products, industrial automation controls, building automation etc.

### 18.2 Features

ESP32-S2 TWAI controller supports the following features:

- compatible with ISO 11898-1 protocol
- Supports Standard Frame Format (11-bit ID) and Extended Frame Format (29-bit ID)
- Bit rates from 1 Kbit/s to 1 Mbit/s
- Multiple modes of operation
  - Normal
  - Listen Only (no influence on bus)
  - Self Test (transmissions do not require acknowledgment)
- 64-byte Receive FIFO
- Special transmissions
  - Single-shot transmissions (does not automatically re-transmit upon error)
  - Self Reception (the TWAI controller transmits and receives messages simultaneously)
- Acceptance Filter (supports single and dual filter modes)
- Error detection and handling
  - Error counters
  - Configurable Error Warning Limit
  - Error Code Capture
  - Arbitration Lost Capture

## 18.3 Functional Protocol

### 18.3.1 TWAI Properties

The TWAI protocol connects two or more nodes in a bus network, and allows for nodes to exchange messages in a latency bounded manner. A TWAI bus will have the following properties.

**Single Channel and Non-Return-to-Zero:** The bus consists of a single channel to carry bits, thus communication is half-duplex. Synchronization is also derived from this channel, thus extra channels (e.g., clock or enable) are not required. The bit stream of a TWAI message is encoded using the Non-Return-to-Zero (NRZ) method.

**Bit Values:** The single channel can either be in a Dominant or Recessive state, representing a logical 0 and a logical 1 respectively. A node transmitting a Dominant state will always override another node transmitting a Recessive state. The physical implementation on the bus is left to the application level to decide (e.g., differential wiring).

**Bit-Stuffing:** Certain fields of TWAI messages are bit-stuffed. A Transmitter that transmits five consecutive bits of the same value should automatically insert a complementary bit. Likewise, a Receiver that receives five consecutive bits should treat the next bit as a stuff bit. Bit stuffing is applied to the following fields: SOF, Arbitration Field, Control Field, Data Field, and CRC Sequence (see Section 18.3.2 for more details).

**Multi-cast:** All nodes receive the same bits as they are connected to the same bus. Data is consistent across all nodes unless there is a bus error (See Section 18.3.3).

**Multi-master:** Any node can initiate a transmission. If a transmission is already ongoing, a node will wait until the current transmission is over before beginning its own transmission.

**Message-Priorities and Arbitration:** If two or more nodes simultaneously initiate a transmission, the TWAI protocol ensures that one node will win arbitration of the bus. The Arbitration Field of the message transmitted by each node is used to determine which node will win arbitration.

**Error Detection and Signaling:** Each node will actively monitor the bus for errors, and signal the detection of errors by transmitting an Error Frame.

**Fault Confinement:** Each node will maintain a set of error counts that are incremented/decremented according to a set of rules. When the error counts surpass a certain threshold, a node will automatically eliminate itself from the network by switching itself off.

**Configurable Bit Rate:** The bit rate for a single TWAI bus is configurable. However, all nodes within the same bus must operate at the same bit rate.

**Transmitters and Receivers:** At any point in time, a TWAI node can either be a Transmitter or a Receiver.

- A node originating a message is a Transmitter. The node remains a Transmitter until the bus is idle or until the node loses arbitration. Note that multiple nodes can be Transmitters if they have yet to lose arbitration.
- All nodes that are not Transmitters are Receivers.

### 18.3.2 TWAI Messages

TWAI nodes use messages to transmit data, and signal errors to other nodes. Messages are split into various frame types, and some frame types will have different frame formats. The TWAI protocol has the following frame types:

- Data Frames

- Remote Frames
- Error Frames
- Overload Frames
- Interframe Space

The TWAI protocol has the following frame formats:

- Standard Frame Format (SFF) that consists of a 11-bit identifier
- Extended Frame Format (EFF) that consists of a 29-bit identifier

### 18.3.2.1 Data Frames and Remote Frames

Data Frames are used by nodes to send data to other nodes, and can have a payload of 0 to 8 data bytes.

Remote Frames are used by nodes to request a Data Frame with the same Identifier from another node, thus does not contain any data bytes. However, Data Frames and Remote Frames share many common fields. Figure 18-1 illustrates the fields and sub fields of the different frames and formats.

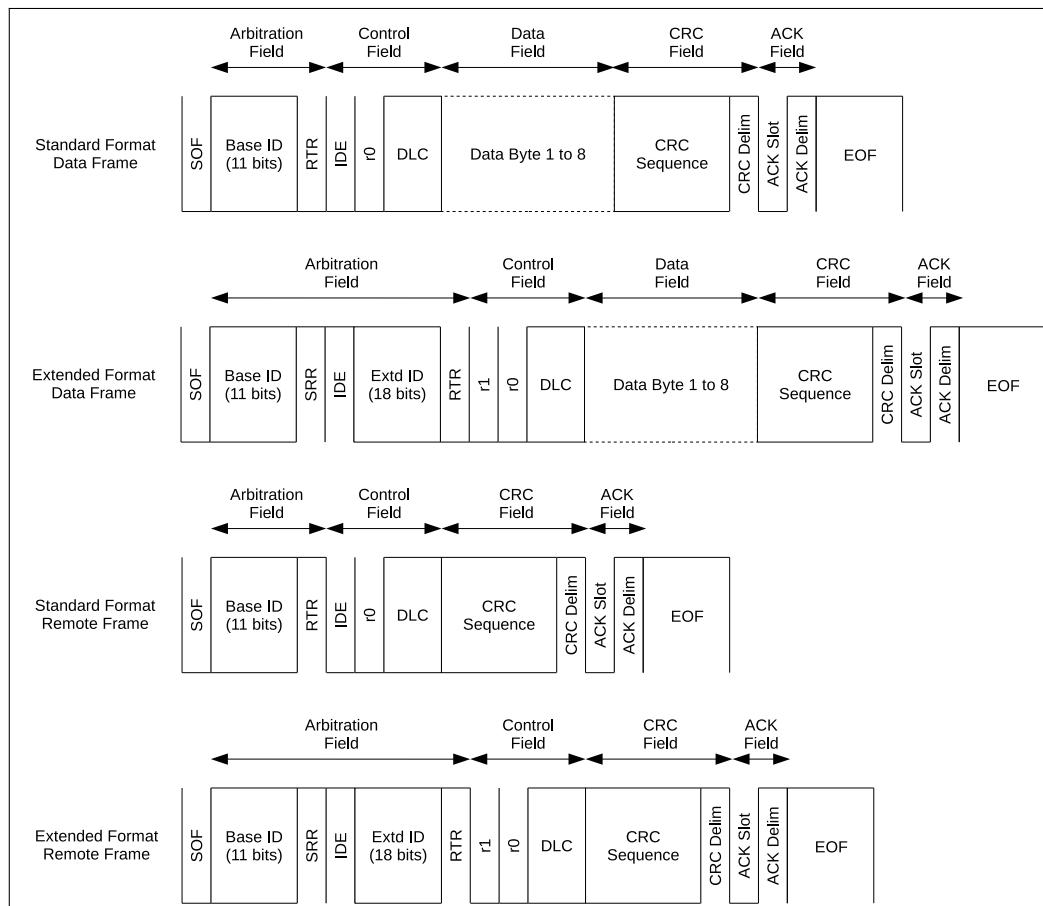


Figure 18-1. The bit fields of Data Frames and Remote Frames

#### Arbitration Field

When two or more nodes transmit a Data or Remote Frame simultaneously, the Arbitration Field is used to determine which node will win arbitration of the bus. During the Arbitration Field, if a node transmits a Recessive bit but observes a Dominant bit, this indicates that another node has overridden its Recessive bit. Therefore, the

node transmitting the Recessive bit has lost arbitration of the bus and should immediately become a Receiver.

The Arbitration Field primarily consists of the Frame Identifier that is transmitted most significant bit first. Given that a Dominant bit represents a logical 0, and a Recessive bit represents a logical 1:

- A frame with the smallest ID value will always win arbitration.
- Given the same ID and format, Data Frames will always prevail over RTR Frames.
- Given the same first 11 bits of ID, a Standard Format Data Frame will prevail over an Extended Format Data Frame due to the SRR being recessive.

### Control Field

The control field primarily consists of the DLC (Data Length Code) which indicates the number of payload data bytes for a Data Frame, or the number of requested data bytes for a Remote Frame. The DLC is transmitted most significant bit first.

### Data Field

The Data Field contains the actual payload data bytes of a Data Frame. Remote Frames do not contain a Data Field.

### CRC Field

The CRC Field primarily consists of a CRC Sequence. The CRC Sequence is a 15-bit cyclic redundancy code calculated from the de-stuffed contents (everything from the SOF to the end of the Data Field) of a Data or Remote Frame.

**Table 68: Data Frames and Remote Frames in SFF and EFF**

Data/Remote Frames	Description
SOF	The SOF (Start of Frame) is a single Dominant bit used to synchronize nodes on the bus.
Base ID	The Base ID (ID.28 to ID.18) is the 11-bit Identifier for SFF, or the first 11-bits of the 29-bit Identifier for EFF.
RTR	The RTR (Remote Transmission Request) bit indicates whether the message is a Data Frame (Dominant) or a Remote Frame (Recessive). This means that a Remote Frame will always lose arbitration to a Data Frame given they have the same ID.
SRR	The SRR (Substitute Remote Request) bit is transmitted in EFF to substitute for the RTR bit at the same position in SFF.
IDE	The IDE (Identifier Extension) bit indicates whether the message is SFF (Dominant) or EFF (Recessive). This means that a SFF frame will always win arbitration over an EFF frame given they have the same Base ID.
Extd ID	The Extended ID (ID.17 to ID.0) is the remaining 18-bits of the 29-bit identifier for EFF.
r1	The r1 (reserved bit 1) is always Dominant.
r0	The r0 (reserved bit 0) is always Dominant.
DLC	The DLC (Data Length Code) is 4-bits and should have a value from 0 to 8. Data Frames use the DLC to indicate the number data bytes in the Data Frame. Remote Frames used the DLC to indicate the number of data bytes to request from another node.

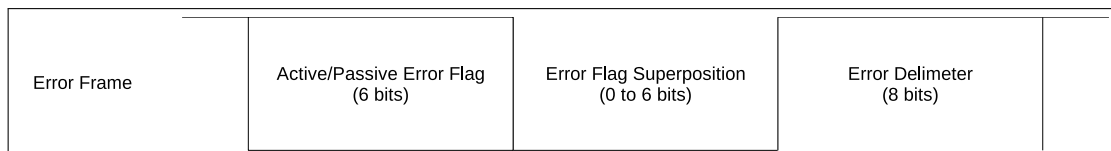


Data/Remote Frames	Description
Data Bytes	The data payload of Data Frames. The number of bytes should match the value of DLC. Data byte 0 is transmitted first, and each data byte is transmitted most significant bit first.
CRC Sequence	The CRC sequence is a 15-bit cyclic redundancy code.
CRC Delim	The CRC Delim (CRC Delimiter) is a single Recessive bit that follows the CRC sequence.
ACK Slot	The ACK Slot (Acknowledgment Slot) that intended for Receiver nodes to indicate that the Data or Remote Frame was received without issue. The Transmitter node will send a Recessive bit in the ACK Slot and Receiver nodes should override the ACK Slot with a Dominant bit if the frame was received without errors.
ACK Delim	The ACK Delim (Acknowledgment Delimiter) is a single Recessive bit.
EOF	The EOF (End of Frame) marks the end of a Data or Remote Frame, and consists of seven Recessive bits.

### 18.3.2.2 Error and Overload Frames

#### Error Frames

Error Frames are transmitted when a node detects a Bus Error. Error Frames notably consist of an Error Flag which is made up of 6 consecutive bits of the same value, thus violating the bit-stuffing rule. Therefore, when a particular node detects a Bus Error and transmits an Error Frame, all other nodes will then detect a Stuff Error and transmit their own Error Frames in response. This has the effect of propagating the detection of a Bus Error across all nodes on the bus. When a node detects a Bus Error, it will transmit an Error Frame starting on the next bit. However, if the type of Bus Error was a CRC Error, then the Error Frame will start at the bit following the ACK Delim (see Section 18.3.3). The following Figure 18-2 shows the various fields of an Error Frame:



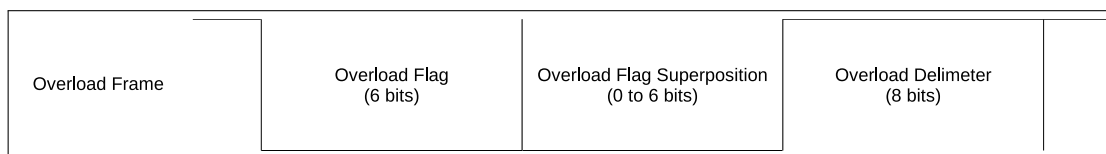
**Figure 18-2. Various Fields of an Error Frame**

**Table 69: Error Frame**

Error Frame	Description
Error Flag	The Error Flag has two forms, the Active Error Flag consisting of 6 Dominant bits and the Passive Error Flag consisting of 6 Recessive bits (unless overridden by Dominant bits of other nodes). Active Error Flags are sent by Error Active nodes, whilst Passive Error Flags are sent by Error Passive nodes.
Error Flag Superposition	The Error Flag Superposition field meant to allow for other nodes on the bus to transmit their respective Active Error Flags. The superposition field can range from 0 to 6 bits, and ends when the first Recessive bit is detected (i.e., the first bit of the Delimiter).
Error Delimiter	The Delimiter field marks the end of the Error/Overload Frame, and consists of 8 Recessive bits.

## Overload Frames

An Overload Frame has the same bit fields as an Error Frame containing an Active Error Flag. The key difference is in the conditions that can trigger the transmission of an Overload Frame. Figure 18-3 below shows the bit fields of an Overload Frame.



**Figure 18-3. The Bit Fields of an Overload Frame**

**Table 70: Overload Frame**

Overload Frame	Description
Overload Flag	Consists of 6 Dominant bits. Same as an Active Error Flag.
Overload Flag Superposition	Allows for the superposition of Overload Flags from other nodes, similar to an Error Flag Superposition.
Overload Delimiter	Consists of 8 Recessive. Same as an Error Delimiter.

Overload Frames will be transmitted under the following conditions:

1. The internal conditions of a Receiver requires a delay of the next Data or Remote Frame.
2. Detection of a Dominant bit at the first and second bit of Intermission.
3. If a Dominant bit is detected at the eighth (last) bit of an Error Delimiter. Note that in this case, TEC and REC will not be incremented (See Section 18.3.3).

Transmitting an overload frame due to one of the conditions must also satisfy the following rules:

- Transmitting an Overload Frame due to condition 1 must only be started at the first bit of Intermission.
- Transmitting an Overload Frame due to condition 2 and 3 must start one bit after the detecting the Dominant bit of the condition.
- A maximum of two Overload frames may be generated in order to delay the next Data or Remote Frame.

### 18.3.2.3 Interframe Space

The Interframe Space acts as a separator between frames. Data Frames and Remote Frames must be separated from preceding frames by an Interframe Space, regardless of the preceding frame's type (Data Frame, Remote Frame, Error Frame, Overload Frame). However, Error Frames and Overload Frames do not need to be separated from preceding frames.

Figure 18-4 shows the fields within an Interframe Space:

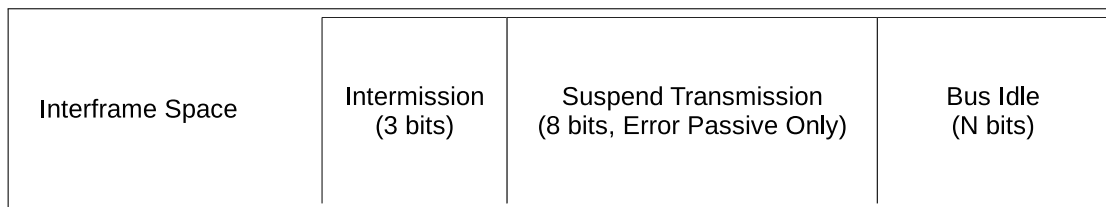


Figure 18-4. The Fields within an Interframe Space

Table 71: Interframe Space

Interface Space	Description
Intermission	The Intermission consists of 3 Recessive bits.
Suspend Trans- mission	An Error Passive node that has just transmitted a message must include a Suspend Transmission field. This field consists of 8 Recessive bits. Error Active nodes should not include this field.
Bus Idle	The Bus Idle field is of arbitrary length. Bus Idle ends when an SOF is transmitted. If a node has a pending transmission, the SOF should be transmitted at the first bit following Intermission.

### 18.3.3 TWAI Errors

#### 18.3.3.1 Error Types

Bus Errors in TWAI are categorized into one of the following types:

##### Bit Error

A Bit Error occurs when a node transmits a bit value (i.e., Dominant or Recessive) but the opposite bit is detected (e.g., a Dominant bit is transmitted but a Recessive is detected). However, if the transmitted bit is Recessive and is located in the Arbitration Field or ACK Slot or Passive Error Flag, then detecting a Dominant bit will not be considered a Bit Error.

##### Stuff Error

A stuff error is detected when 6 consecutive bits of the same value are detected (thus violating the bit-stuffing encoding).

##### CRC Error

A Receiver of a Data or Remote Frame will calculate a CRC based on the bits it has received. A CRC error occurs when the CRC calculated by the Receiver does not match the CRC sequence in the received Data or Remote Frame.

##### Form Error

A Form Error is detected when a fixed-form bit field of a message contains an illegal bit. For example, the r1 and r0 fields must be Dominant.

##### Acknowledgement Error

An Acknowledgment Error occurs when a Transmitter does not detect a Dominant bit at the ACK Slot.

#### 18.3.3.2 Error States

TWAI nodes implement fault confinement by each maintaining two error counters, where the counter values determine the error state. The two error counters are known as the Transmit Error Counter (TEC) and Receive Error Counter (REC). TWAI has the following error states.

**Error Active**

An Error Active node is able to participate in bus communication and transmit an Active Error Flag when it detects an error.

**Error Passive**

An Error Passive node is able to participate in bus communication, but can only transmit a Passive Error Flag when it detects an error. Error Passive nodes that have transmitted a Data or Remote Frame must also include the Suspend Transmission field in the subsequent Interframe Space.

**Bus Off**

A Bus Off node is not permitted to influence the bus in any way (i.e., is not allowed to transmit anything).

### 18.3.3.3 Error Counters

The TEC and REC are incremented/decremented according to the following rules. **Note that more than one rule can apply for a given message transfer.**

1. When a Receiver detects an error, the REC will be increased by 1, except when the detected error was a Bit Error during the transmission of an Active Error Flag or an Overload Flag.
2. When a Receiver detects a Dominant bit as the first bit after sending an Error Flag, the REC will be increased by 8.
3. When a Transmitter sends an Error Flag the TEC is increased by 8. However, the following scenarios are exempt from this rule:
  - If a Transmitter is Error Passive that detects an Acknowledgment Error due to not detecting a Dominant bit in the ACK slot, it should send a Passive Error Flag. If no Dominant bit is detected in that Passive Error Flag, the TEC should not be increased.
  - A Transmitter transmits an Error Flag due to a Stuff Error during Arbitration. If the offending bit should have been Recessive but was monitored as Dominant, then the TEC should not be increased.
4. If a Transmitter detects a Bit Error whilst sending an Active Error Flag or Overload Flag, the REC is increased by 8.
5. If a Receiver detects a Bit Error while sending an Active Error Flag or Overload Flag, the REC is increased by 8.
6. Any node tolerates up to 7 consecutive Dominant bits after sending an Active/Passive Error Flag, or Overload Flag. After detecting the 14th consecutive Dominant bit (when sending an Active Error Flag or Overload Flag), or the 8th consecutive Dominant bit following a Passive Error Flag, a Transmitter will increase its TEC by 8 and a Receiver will increase its REC by 8. Each additional eight consecutive Dominant bits will also increase the TEC (for Transmitters) or REC (for Receivers) by 8 as well.
7. When a Transmitter successfully transmits a message (getting ACK and no errors until the EOF is complete), the TEC is decremented by 1, unless the TEC is already at 0.
8. When a Receiver successfully receives a message (no errors before ACK Slot, and successful sending of ACK), the REC is decremented.
  - If the REC was between 1 and 127, the REC is decremented by 1.
  - If the REC was greater than 127, the REC is set to 127.

- If the REC was 0, the REC remains 0.
- A node becomes Error Passive when its TEC and/or REC is greater than or equal to 128. The error condition that causes a node to become Error Passive will cause the node to send an Active Error Flag. Note that once the REC has reached to 128, any further increases to its value are irrelevant until the REC returns to a value less than 128.
  - A node becomes Bus Off when its TEC is greater than or equal to 256.
  - An Error Passive node becomes Error Active when both the TEC and REC are less than or equal to 127.
  - A Bus Off node can become Error Active (with both its TEC and REC reset to 0) after it monitors 128 occurrences of 11 consecutive Recessive bits on the bus.

### 18.3.4 TWAI Bit Timing

#### 18.3.4.1 Nominal Bit

The TWAI protocol allows a TWAI bus to operate at a particular bit rate. However, all nodes within a TWAI bus must operate at the same bit rate.

- The **Nominal Bit Rate** is defined as number of bits transmitted per second from an ideal Transmitter and without any synchronization.
- The **Nominal Bit Time** is defined as  $1/\text{Nominal Bit Rate}$ .

A single Nominal Bit Time is divided into multiple segments, and each segment is made up of multiple Time Quanta. A **Time Quantum** is a fixed unit of time, and is implemented as some form of prescaled clock signal in each node. Figure 18-5 illustrates the segments within a single Nominal Bit Time.

TWAI Controllers will operate in time steps of one Time Quanta where the state of the TWAI bus is analyzed at every Time Quanta. If two consecutive Time Quantas have different bus states (i.e., Recessive to Dominant or vice versa), this will be considered an edge. When the bus is analyzed at the intersection of PBS1 and PBS2, this is considered the Sample Point and the sampled bus value is considered the value of that bit.

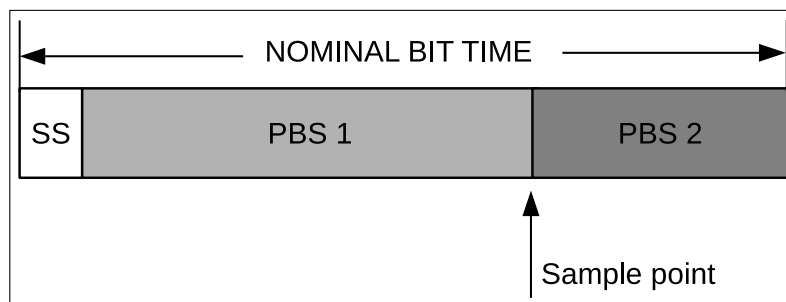


Figure 18-5. Layout of a Bit

Table 72: Segments of a Nominal Bit Time

Segment	Description
SS	The SS (Synchronization Segment) is 1 Time Quantum long. If all nodes are perfectly synchronized, the edge of a bit will lie in the SS.
PBS1	PBS1 (Phase Buffer Segment 1) can be 1 to 16 Time Quanta long. PBS1 is meant to compensate for the physical delay times within the network. PBS1 can also be lengthened for synchronization purposes.

Segment	Description
PBS2	PBS2 (Phase Buffer Segment 2) can be 1 to 8 Time Quanta long. PBS2 is meant to compensate for the information processing time of nodes. PBS2 can also be shortened for synchronization purposes.

#### 18.3.4.2 Hard Synchronization and Resynchronization

Due to clock skew and jitter, the bit timing of nodes on the same bus may become out of phase. Therefore, a bit edge may come before or after the SS. To ensure that the internal bit timing clocks of each node are kept in phase, TWAI has various methods of synchronization. The **Phase Error “e”** is measured in the number of Time Quanta and relative to the SS.

- A positive Phase Error ( $e > 0$ ) is when the edge lies after the SS and before the Sample Point (i.e., the edge is late).
- A negative Phase Error ( $e < 0$ ) is when the edge lies after the Sample Point of the previous bit and before SS (i.e., the edge is early).

To correct for Phase Errors, there are two forms of synchronization, known as **Hard Synchronization** and **Resynchronization**. **Hard Synchronization** and **Resynchronization** obey the following rules.

- Only one synchronization may occur in a single bit time.
- Synchronizations only occurs on Recessive to Dominant edges.

##### Hard Synchronization

Hard Synchronization occurs on the Recessive to Dominant edges during Bus Idle (i.e., the SOF bit). All nodes will restart their internal bit timings such that the Recessive to Dominant edge lies within the SS of the restarted bit timing.

##### Resynchronization

Resynchronization occurs on Recessive to Dominant edges not during Bus Idle. If the edge has a positive Phase Error ( $e > 0$ ), PBS1 is lengthened by a certain number of Time Quanta. If the edge has a negative Phase Error ( $e < 0$ ), PBS2 will be shortened by a certain number of Time Quanta.

The number of Time Quanta to lengthen or shorten depends on the magnitude of the Phase Error, and is also limited by the Synchronization Jump Width (SJW) value which is a programmable.

- When the magnitude of the Phase Error is less than or equal to the SJW, PBS1/PBS2 are lengthened/shortened by **e** number of Time Quanta. This has a same effect as Hard Synchronization.
- When the magnitude of the Phase Error is greater to the SJW, PBS1/PBS2 are lengthened/shortened by the SJW number of Time Quanta. This means it may take multiple bits of synchronization before the Phase Error is entirely corrected.

## 18.4 Architectural Overview

The ESP32-S2 contains a TWAI Controller. Figure 18-6 show show the major functional blocks of the TWAI Controller.

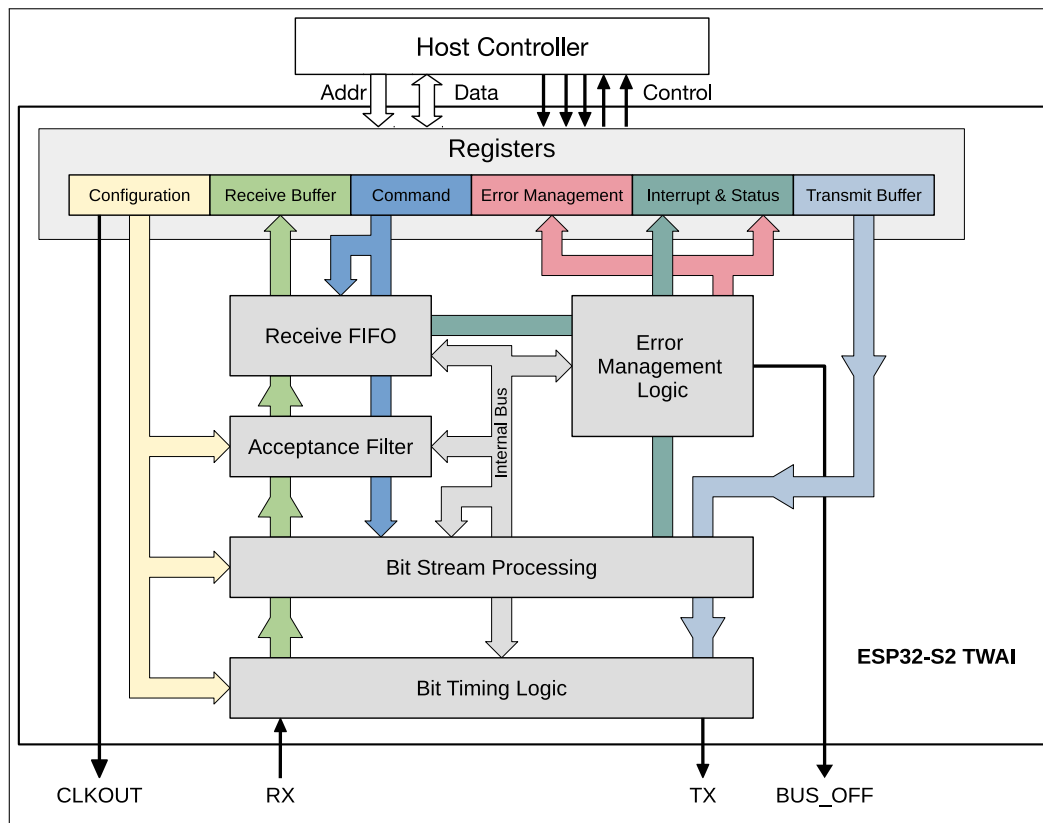


Figure 18-6. TWAI Overview Diagram

### 18.4.1 Registers Block

The ESP32-S2 CPU accesses peripherals as 32-bit aligned words. However, the majority of registers in the TWAI controller only contain useful data at the least significant byte (bits [7:0]). Therefore, in these registers, bits [31:8] are ignored on writes, and return 0 on reads.

#### Configuration Registers

The configuration registers store various configuration options for the TWAI controller such as bit rates, operating mode, Acceptance Filter etc. Configuration registers can only be modified whilst the TWAI controller is in Reset Mode (See Section 18.5.1).

#### Command Register

The command register is used by the CPU to drive the TWAI controller to initiate certain actions such as transmitting a message or clearing the Receive Buffer. The command register can only be modified when the TWAI controller is in Operation Mode (see section 18.5.1).

#### Interrupt & Status Registers

The interrupt register indicates what events have occurred in the TWAI controller (each event is represented by a separate bit). The status register indicates the current status of the TWAI controller.

#### Error Management Registers

The error management registers include error counters and capture registers. The error counter registers represent TEC and REC values. The capture registers will record information about instances where TWAI controller detects a bus error, or when it loses arbitration.

#### Transmit Buffer Registers

The transmit buffer is a 13-byte buffer used to store a TWAI message to be transmitted.

### Receive Buffer Registers

The Receive Buffer is a 13-byte buffer which stores a single message. The Receive Buffer acts as a window into Receive FIFO mapping to the first received message in the Receive FIFO to the Receive Buffer.

Note that the Transmit Buffer registers, Receive Buffer registers, and the Acceptance Filter registers share the same address range (offset 0x0040 to 0x0070). Their access is governed by the following rules:

- When the TWAI controller is in Reset Mode, the address range maps to the Acceptance Filter registers.
- When the TWAI controller is in Operation Mode:
  - All reads to the address range maps to the Receive Buffer registers.
  - All writes to the address range maps to the Transmit Buffer registers.

### 18.4.2 Bit Stream Processor

The Bit Stream Processing (BSP) module is responsible for framing data from the Transmit Buffer (e.g. bit stuffing and additional CRC fields) and generating a bit stream for the Bit Timing Logic (BTL) module. At the same time, the BSP module is also responsible for processing the received bit stream (e.g., de-stuffing and verifying CRC) from the BTL module and placing the message into the Receive FIFO. The BSP will also detect errors on the TWAI bus and report them to the Error Management Logic (EML).

### 18.4.3 Error Management Logic

The Error Management Logic (EML) module is responsible for updating the TEC and REC, recording error information like error types and positions, and updating the error state of the TWAI Controller such that the BSP module generates the correct Error Flags. Furthermore, this module also records the bit position when the TWAI controller loses arbitration.

### 18.4.4 Bit Timing Logic

The Bit Timing Logic (BTL) module is responsible for transmitting and receiving messages at the configured bit rate. The BTL module also handles synchronization of out of phase bits such that communication remains stable. A single bit time consists of multiple programmable segments that allows users to set the length of each segment to account for factors such as propagation delay and controller processing time etc.

### 18.4.5 Acceptance Filter

The Acceptance Filter is a programmable message filtering unit that allows the TWAI controller to accept or reject a received message based on the message's ID field. Only accepted messages will be stored in the Receive FIFO. The Acceptance Filter's registers can be programmed to specify a single filter, or specify two separate filters (dual filter mode).

### 18.4.6 Receive FIFO

The Receive FIFO is a 64-byte buffer (internal to the TWAI controller) that stores received messages accepted by the Acceptance Filter. Messages in the Receive FIFO can vary in size (between 3 to 13-bytes). When the Receive FIFO is full (or does not have enough space to store the next received message in its entirety), the Overrun Interrupt will be triggered, and any subsequent received messages will be lost until adequate space is cleared in the Receive FIFO. The first message in the Receive FIFO will be mapped to the 13-byte Receive Buffer until that message is cleared (using the Release Receive Buffer command bit). After clearing, the Receive Buffer will map to the next message in the Receive FIFO, and the space occupied by the previous message in the Receive FIFO can be used to receive new messages.



## 18.5 Functional Description

### 18.5.1 Modes

The ESP32-S2 TWAI controller has two working modes: Reset Mode and Operation Mode. Reset Mode and Operation Mode are entered by setting the [TWAI\\_RESET\\_MODE](#) bit to 1 or 0 respectively.

#### 18.5.1.1 Reset Mode

Entering Reset Mode is required in order to modify the various configuration registers of the TWAI controller. When entering Reset Mode, the TWAI controller is essentially disconnected from the TWAI bus. When in Reset Mode, the TWAI controller will not be able to transmit any messages (including error signaling). Any transmission in progress is immediately terminated. Likewise, the TWAI controller will also not be able to receive any messages.

#### 18.5.1.2 Operation Mode

Entering Operation Mode essentially connects the TWAI controller to the TWAI bus, and write protects the TWAI controller's configuration registers ensuring the configuration stays consistent during operation. When in Operation Mode, the TWAI controller can transmit and receive messages (including error signaling) depending on which operating sub-mode the TWAI controller was configured with. The TWAI controller supports the following operating sub-modes:

- **Normal Mode:** The TWAI controller can transmit and receive messages including error signaling (such as Error and Overload Frames).
- **Self Test Mode:** Like Normal Mode, but the TWAI controller will consider the transmission of a Data or RTR Frame successful even if it was not acknowledged. This is commonly used when self testing the TWAI controller.
- **Listen Only Mode:** The TWAI controller will be able to receive messages, but will remain completely passive on the TWAI bus. Thus, the TWAI controller will not be able to transmit any messages, acknowledgments, or error signals. The error counters will remain frozen. This mode is useful for TWAI bus monitors.

Note that when exiting Reset Mode (i.e., entering Operation Mode), the TWAI controller must wait for 11 consecutive Recessive bits to occur before being able to fully connect the TWAI bus (i.e., be able to transmit or receive).

### 18.5.2 Bit Timing

The operating bit rate of the TWAI controller must be configured whilst the TWAI controller is in Reset Mode. The bit rate configuration is located in [TWAI\\_BUS\\_TIMING\\_0\\_REG](#) and [TWAI\\_BUS\\_TIMING\\_1\\_REG](#), and the two registers contain the following fields:

The following Table 73 illustrates the bit fields of [TWAI\\_BUS\\_TIMING\\_0\\_REG](#).

**Table 73: Bit Information of [TWAI\\_CLOCK\\_DIVIDER\\_REG](#); TWAI Address 0x18**

Bit 31-16	Bit 15	Bit 14	Bit 13	Bit 12	.....	Bit 1	Bit 0
Reserved	SJW.1	SJW.0	BRP.13	BRP.12	.....	BRP.1	BRP.0

#### Notes:

- SJW: Synchronization Jump Width (SJW) is configured in SJW.0 and SJW.1 where  $SJW = (2 \times SJW.1 + SJW.0 + 1)$ .
- BRP: The TWAI Time Quanta clock is derived from a prescaled version of the APB clock that is usually 80 MHz. The Baud Rate Prescaler (BRP) field is used to define the prescaler according to the equation below, where  $t_{Tq}$  is the Time Quanta clock period and  $t_{CLK}$  is APB clock period :  

$$t_{Tq} = 2 \times t_{CLK} \times (2^{13} \times BRP.13 + 2^{12} \times BRP.12 + \dots + 2^1 \times BRP.1 + 2^0 \times BRP.0 + 1)$$

The following Table 74 illustrates the bit fields of [TWAI\\_BUS\\_TIMING\\_1\\_REG](#).

**Table 74: Bit Information of [TWAI\\_BUS\\_TIMING\\_1\\_REG](#); TWAI Address 0x1c**

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	SAM	PBS2.2	PBS2.1	PBS2.0	PBS1.3	PBS1.2	PBS1.1	PBS1.0

**Notes:**

- PBS1: The number of Time Quanta in Phase Buffer Segment 1 is defined according to the following equation:  $(8 \times PBS1.3 + 4 \times PBS1.2 + 2 \times PBS1.1 + PBS1.0 + 1)$ .
- PBS2: The number of Time Quanta in Phase Buffer Segment 2 is defined according to the following equation:  $(4 \times PBS2.2 + 2 \times PBS2.1 + PBS2.0 + 1)$ .
- SAM: Enables triple sampling if set to 1. This is useful for low/medium speed buses where filtering spikes on the bus line is beneficial.

### 18.5.3 Interrupt Management

The ESP32-S2 TWAI controller provides seven interrupts, each represented by a single bit in the [TWAI\\_INT\\_RAW\\_REG](#). For a particular interrupt to be triggered (i.e., its bit in [TWAI\\_INT\\_RAW\\_REG](#) set to 1), the interrupt's corresponding enable bit in [TWAI\\_INT\\_ENA\\_REG](#) must be set.

The TWAI controller provides the seven following interrupts:

- Receive Interrupt
- Transmit Interrupt
- Error Warning Interrupt
- Data Overrun Interrupt
- Error Passive Interrupt
- Arbitration Lost Interrupt
- Bus Error Interrupt

The TWAI controller's interrupt signal to the interrupt matrix will be asserted whenever one or more interrupt bits are set in the [TWAI\\_INT\\_RAW\\_REG](#), and deasserted when all bits in [TWAI\\_INT\\_RAW\\_REG](#) are cleared. The majority of interrupt bits in [TWAI\\_INT\\_RAW\\_REG](#) are automatically cleared when the register is read. However, the Receive Interrupt is an exception and can only be cleared the Receive FIFO is empty.

#### 18.5.3.1 Receive Interrupt (RXI)

The Receive Interrupt (RXI) is asserted whenever the TWAI controller has received messages that are pending to read from the Receive Buffer (i.e., when [TWAI\\_RX\\_MESSAGE\\_CNT\\_REG](#) > 0). Pending received messages

includes valid messages in the Receive FIFO and also overrun messages. The RXI will not be deasserted until all pending received messages are cleared using the [TWAI\\_RELEASE\\_BUF](#) command bit.

### 18.5.3.2 Transmit Interrupt (TXI)

The Transmit Interrupt (TXI) is triggered whenever Transmit Buffer becomes free, indicating another message can be loaded into the Transmit Buffer to be transmitted. The Transmit Buffer becomes free under the following scenarios:

- A message transmission has completed successfully (i.e., Acknowledged without any errors). Any failed messages will automatically be retried.
- A single shot transmission has completed (successfully or unsuccessfully, indicated by the [TWAI\\_TX\\_COMPLETE](#) bit).
- A message transmission was aborted using the [TWAI\\_ABORT\\_TX](#) command bit.

### 18.5.3.3 Error Warning Interrupt (EWI)

The Error Warning Interrupt (EWI) is triggered whenever there is a change to the [TWAI\\_ERR\\_ST](#) or [TWAI\\_BUS\\_OFF\\_ST](#) bits of the [TWAI\\_STATUS\\_REG](#) (i.e., transition from 0 to 1 or vice versa). Thus, an EWI could indicate one of the following events, depending on the values [TWAI\\_ERR\\_ST](#) or [TWAI\\_BUS\\_OFF\\_ST](#) at the moment the EWI is triggered.

- If [TWAI\\_ERR\\_ST](#) = 0 or [TWAI\\_BUS\\_OFF\\_ST](#) = 0:
  - If the TWAI controller was in the Error Active state, it indicates both the TEC and REC have returned below the threshold value set by [TWAI\\_ERR\\_WARNING\\_LIMIT\\_REG](#).
  - If the TWAI controller was previously in the Bus Recovery state, it indicates that Bus Recovery has completed successfully.
- If [TWAI\\_ERR\\_ST](#) = 1 or [TWAI\\_BUS\\_OFF\\_ST](#) = 0: The TEC or REC error counters have exceeded the threshold value set by [TWAI\\_ERR\\_WARNING\\_LIMIT\\_REG](#).
- If [TWAI\\_ERR\\_ST](#) = 1 or [TWAI\\_BUS\\_OFF\\_ST](#) = 1: The TWAI controller has entered the BUS\_OFF state (due to the TEC >= 256).
- If [TWAI\\_ERR\\_ST](#) = 0 or [TWAI\\_BUS\\_OFF\\_ST](#) = 1: The TWAI controller's TEC has dropped below the threshold value set by [TWAI\\_ERR\\_WARNING\\_LIMIT\\_REG](#) during BUS\_OFF recovery.

### 18.5.3.4 Data Overrun Interrupt (DOI)

The Data Overrun Interrupt (DOI) is triggered whenever the Receive FIFO has overrun. The DOI indicates that the Receive FIFO is full and should be cleared immediately to prevent any further overrun messages.

The DOI is only triggered on the first message that causes the Receive FIFO to overrun (i.e., the transition from the Receive FIFO not being full to the Receive FIFO overflowing). Any subsequent overrun messages will not trigger the DOI again. The DOI will only be able to trigger again when all received messages (valid or overrun) have been cleared.

### 18.5.3.5 Error Passive Interrupt (TXI)

The Error Passive Interrupt (EPI) is triggered whenever the TWAI controller transitions from Error Active to Error Passive, or vice versa.

### 18.5.3.6 Arbitration Lost Interrupt (ALI)

The Arbitration Lost Interrupt (ALI) is triggered whenever the TWAI controller is attempting to transmit a message and loses arbitration. The bit position where the TWAI controller lost arbitration is automatically recorded in Arbitration Lost Capture register ([TWAI\\_ARB\\_LOST\\_CAP\\_REG](#)). When the ALI occurs again, the Arbitration Lost Capture register will no longer record new bit location until it is cleared (via a read from the CPU).

### 18.5.3.7 Bus Error Interrupt (BEI)

The Bus Error Interrupt (BEI) is triggered whenever TWAI controller observes an error on the TWAI bus. When a bus error occurs, the Bus Error type and its bit position are automatically recorded in the Error Code Capture register ([TWAI\\_ERR\\_CODE\\_CAP\\_REG](#)). When the BEI occurs again, the Error Code Capture register will no longer record new error information until it is cleared (via a read from the CPU).

## 18.5.4 Transmit and Receive Buffers

### 18.5.4.1 Overview of Buffers

**Table 75: Buffer Layout for Standard Frame Format and Extended Frame Format**

Standard Frame Format (SFF)		Extended Frame Format (EFF)	
TWAI address	Content	TWAI address	Content
0x40	TX/RX frame information	0x40	TX/RX frame information
0x44	TX/RX identifier 1	0x44	TX/RX identifier 1
0x48	TX/RX identifier 2	0x48	TX/RX identifier 2
0x4c	TX/RX data byte 1	0x4c	TX/RX identifier 3
0x50	TX/RX data byte 2	0x50	TX/RX identifier 4
0x54	TX/RX data byte 3	0x54	TX/RX data byte 1
0x58	TX/RX data byte 4	0x58	TX/RX data byte 2
0x5c	TX/RX data byte 5	0x5c	TX/RX data byte 3
0x60	TX/RX data byte 6	0x60	TX/RX data byte 4
0x64	TX/RX data byte 7	0x64	TX/RX data byte 5
0x68	TX/RX data byte 8	0x68	TX/RX data byte 6
0x6c	reserved	0x6c	TX/RX data byte 7
0x70	reserved	0x70	TX/RX data byte 8

Table 75 illustrates the layout of the Transmit Buffer and Receive Buffer registers. Both the Transmit and Receive Buffer registers share the same address space and are only accessible when the TWAI controller is in Operation Mode. CPU write operations will access the Transmit Buffer registers, and CPU read operations will access the Receive Buffer registers. However, both buffers share the exact same register layout and fields to represent a message (received or to be transmitted). The Transmit Buffer registers are used to configure a TWAI message to be transmitted. The CPU would write to the Transmit Buffer registers specifying the message's frame type, frame

format, frame ID, and frame data (payload). Once the Transmit Buffer is configured, the CPU would then initiate the transmission by setting the [TWAI\\_TX\\_REQ](#) bit in [TWAI\\_CMD\\_REG](#).

- For a self-reception request, set the [TWAI\\_SELF\\_RX\\_REQ](#) bit instead.
- For a single-shot transmission, set both the [TWAI\\_TX\\_REQ](#) and the [TWAI\\_ABORT\\_TX](#) simultaneously.

The Receive Buffer registers map to the first message in the Receive FIFO. The CPU would read the Receive Buffer registers to obtain the first message's frame type, frame format, frame ID, and frame data (payload). Once the message has been read from the Receive Buffer registers, the CPU can set the [TWAI\\_RELEASE\\_BUF](#) bit in [TWAI\\_CMD\\_REG](#) so that the next message in the Receive FIFO will be loaded in to the Receive Buffer registers.

### 18.5.4.2 Frame Information

The frame information is one byte long and specifies a message's frame type, frame format, and length of data. The frame information fields are shown in Table 76.

**Table 76: TX/RX Frame Information (SFF/EFF) TWAI Address 0x40**

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	FF <sup>1</sup>	RTR <sup>2</sup>	X <sup>3</sup>	X <sup>3</sup>	XDLC.3 <sup>4</sup>	DLC.2 <sup>4</sup>	DLC.1 <sup>4</sup>	DLC.0 <sup>4</sup>

#### Notes:

- FF: The Frame Format (FF) bit specifies whether the message is Extended Frame Format (EFF) or Standard Frame Format (SFF). The message is EFF when FF bit is 1, and SFF when FF bit is 0.
- RTR: The Remote Transmission Request (RTR) bit specifies whether the message is a Data Frame or a Remote Frame. The message is a Remote Frame when the RTR bit is 1, and a Data Frame when the RTR bit is 0.
- DLC: The Data Length Code (DLC) field specifies the number of data bytes for a Data Frame, or the number of data bytes to request in a Remote Frame. TWAI Data Frames are limited to a maximum payload of 8 data bytes, thus the DLC should range anywhere from 0 to 8.
- X: Don't care, can be any value.

### 18.5.4.3 Frame Identifier

The Frame Identifier fields is 2 bytes (11-bits) if the message is SFF, and 4 bytes (29-bits) if the message is EFF.

The Frame Identifier fields for an SFF (11-bits) message is shown in Table 77-78.

**Table 77: TX/RX Identifier 1 (SFF); TWAI Address 0x44**

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21

**Table 78: TX/RX Identifier 2 (SFF); TWAI Address 0x48**

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	ID.20	ID.19	ID.18	X <sup>1</sup>	X <sup>2</sup>	X <sup>2</sup>	X <sup>2</sup>	X <sup>2</sup>

The Frame Identifier fields for an EFF (29-bits) message is shown in Table 79-82.

**Table 79: TX/RX Identifier 1 (EFF); TWAI Address 0x44**

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21

**Table 80: TX/RX Identifier 2 (EFF); TWAI Address 0x48**

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13

**Table 81: TX/RX Identifier 3 (EFF); TWAI Address 0x4c**

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5

**Table 82: TX/RX Identifier 4 (EFF); TWAI Address 0x50**

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	ID.4	ID.3	ID.2	ID.1	ID.0	X <sup>1</sup>	X <sup>2</sup>	X <sup>2</sup>

#### 18.5.4.4 Frame Data

The Frame Data fields contains the payload of transmitted or received a Data Frame, and can range from 0 to 8 bytes. The number of valid bytes should be equal to the DLC. However, if the DLC is larger than 8, the number of valid bytes would still be limited to 8. Remote Frames do not have data payloads, thus the Frame Data fields will be unused.

For example, when transmitting a Data Frame with 5 data bytes, the CPU should write a value of 5 to the DLC field, and then fill in data bytes 1 to 5 in the Frame Data fields. Likewise, when receiving a Data Frame with a DLC of 5, only data bytes 1 to 5 will contain valid payload data for the CPU to read.

#### 18.5.5 Receive FIFO and Data Overruns

The Receive FIFO is a 64-byte internal buffer used to store received messages in First In First Out order. A single received message can occupy between 3 to 13-bytes of space in the Receive FIFO, and their byte layout is identical to the register layout of the Receive Buffer registers. The Receive Buffer registers are mapped to the bytes of the first message in the Receive FIFO. When the TWAI controller receives a message, it will increment the value of [TWAI\\_RX\\_MESSAGE\\_COUNTER](#) up to a maximum of 64. If there is adequate space in the Receive FIFO, the message contents will be written into the Receive FIFO. Once a message has been read from the Receive Buffer, the [TWAI\\_RELEASE\\_BUF](#) bit should be set. This will decrement [TWAI\\_RX\\_MESSAGE\\_COUNTER](#) and free the space occupied by the first message in the Receive FIFO. The

Receive Buffer will then map to the next message in the Receive FIFO. A data overrun occurs when the TWAI controller receives a message, but the Receive FIFO lacks the adequate free space to store the received message in its entirety (either due to the message contents being larger than the free space in the Receive FIFO, or the Receive FIFO being completely full).

When a data overrun occurs...

- Whatever free space is left in the Receive FIFO is filled with the partial contents of the overrun message. If the Receive FIFO is already full, then none of the overrun message's contents will be stored.
- On the first message that causes the Receive FIFO to overrun, a Data Overrun Interrupt will be triggered.
- Each overrun message will still increment the [TWAI\\_RX\\_MESSAGE\\_COUNTER](#) up to a maximum of 64.
- The RX FIFO will internally mark overrun messages as invalid. The [TWAI\\_MISS\\_ST](#) bit can be used to determine whether the message currently mapped to by the Receive Buffer is valid or overrun.

To clear an overrun Receive FIFO, the [TWAI\\_RELEASE\\_BUF](#) must be called repeatedly until [TWAI\\_RX\\_MESSAGE\\_COUNTER](#) is 0. This has the effect of freeing all valid messages in the Receive FIFO and clearing all overrun messages.

### 18.5.6 Acceptance Filter

The Acceptance Filter allows the TWAI controller to filter out received messages based on their ID (and optionally their first data byte and frame type). Only accepted messages are passed on to the Receive FIFO. The use of Acceptance Filters allows for a more lightweight operation of the TWAI controller (e.g., less use of Receive FIFO, fewer Receive Interrupts) due to the TWAI Controller only needing to handle a subset of messages.

The Acceptance Filter configuration registers can only be accessed whilst the TWAI controller is in Reset Mode, due to those registers sharing the same address space as the Transmit Buffer and Receive Buffer registers.

The registers consist of a 32-bit Acceptance Code Value and a 32-bit Acceptance Mask Value. The Code value specifies a bit pattern in which each filtered bit of the message must match in order for the message to be accepted. The Mask value is able to mask out certain bits of the Code value (i.e., set as "Don't Care" bits). Each filtered bit of the message must either match the acceptance code or be masked in order for the message to be accepted, as demonstrated in Figure 18-7.

The TWAI Controller Acceptance Filter allows the 32-bit Code and Mask values to either define a single filter (i.e., Single Filter Mode), or two filters (i.e., Dual Filter Mode). How the Acceptance Filter interprets the 32-bit code and mask values is dependent on whether Single Filter Mode is enabled, and the received message (i.e., SFF or EFF).

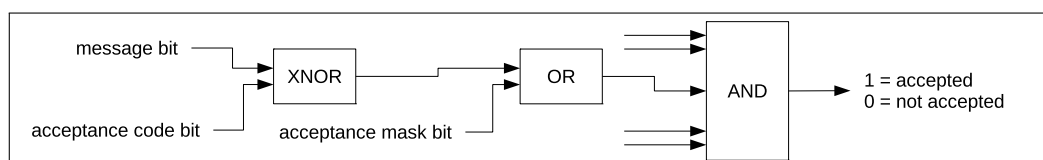


Figure 18-7. Acceptance Filter

#### 18.5.6.1 Single Filter Mode

Single Filter Mode is enabled by setting the [TWAI\\_RX\\_FILTER\\_MODE](#) bit to 1. This will cause the 32-bit code and mask values to define a single filter. The single filter can filter the following bits of a Data or Remote Frame:

- SFF
  - The entire 11-bit ID
  - RTR bit
  - Data byte 1 and Data byte 2
- EFF
  - The entire 29-bit ID
  - RTR bit

The following Figure 18-8 illustrates how the 32-bit code and mask values will be interpreted under Single Filter Mode.

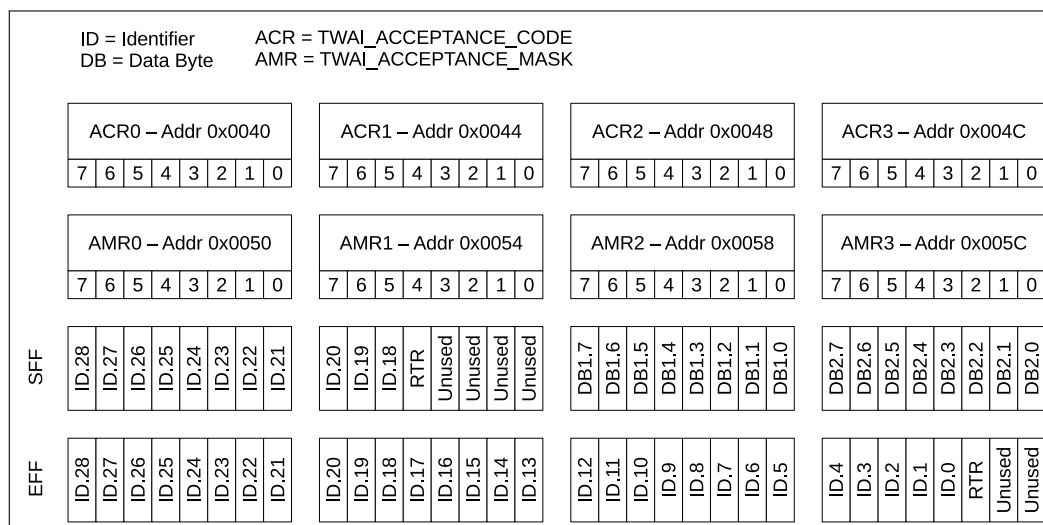


Figure 18-8. Single Filter Mode

### 18.5.6.2 Dual Filter Mode

Dual Filter Mode is enabled by setting the `TWAI_RX_FILTER_MODE` bit to 0. This will cause the 32-bit code and mask values to define a two separate filters, referred to as filter 1 or two. Under Dual Filter Mode, a message will be accepted if it is accepted by one of the two filters.

The two filters can filter the following bits of a Data or Remote Frame:

- SFF
  - The entire 11-bit ID
  - RTR bit
  - Data byte 1 (for filter 1 only)
- EFF
  - The first 16 bits of the 29-bit ID

The following Figure 18-9 illustrates how the 32-bit code and mask values will be interpreted under Dual Filter Mode.



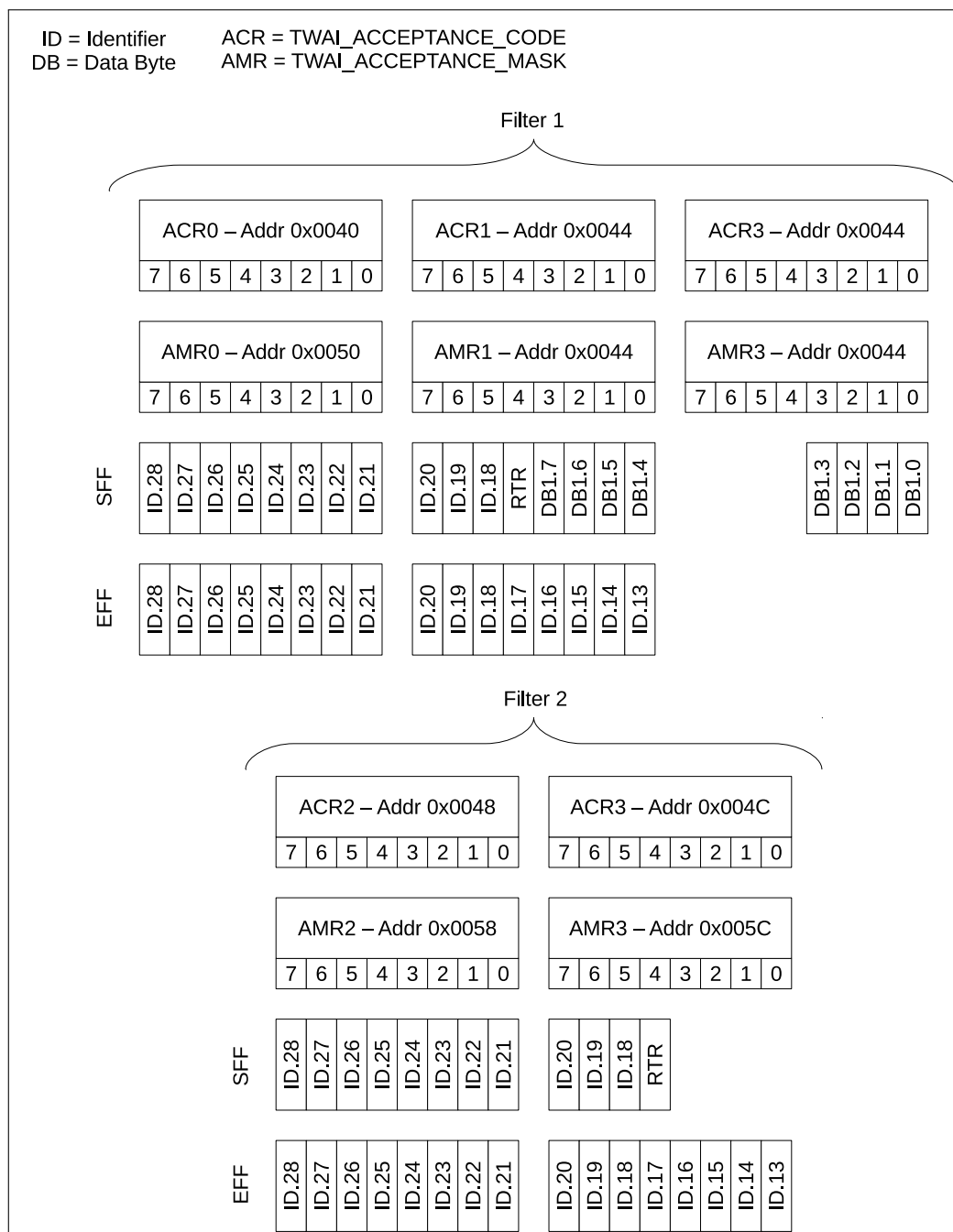


Figure 18-9. Dual Filter Mode

### 18.5.7 Error Management

The TWAI protocol requires that each TWAI node maintains the Transmit Error Count (TEC) and Receive Error Count (REC). The value of both error counts determine the current error state of the TWAI controller (i.e., Error Active, Error Passive, Bus-Off). The TWAI controller stores the TEC and REC values in the [TWAI\\_TX\\_ERR\\_CNT\\_REG](#) and [TWAI\\_RX\\_ERR\\_CNT\\_REG](#) respectively, and can be read by the CPU at anytime. In addition to the error states, the TWAI controller also offers an Error Warning Limit (EWL) feature that can warn the user regarding the occurrence of severe bus errors before the TWAI controller enters the Error Passive state.

The current error state of the TWAI controller is indicated via a combination of the following values and status bits: TEC, REC, [TWAI\\_ERR\\_ST](#), and [TWAI\\_BUS\\_OFF\\_ST](#). Certain changes to these values and bits will also trigger

interrupts, thus allowing the users to be notified of error state transitions (see section 18.5.3). The following figure 18-10 shows the relation between the error states, values and bits, and error state related interrupts.

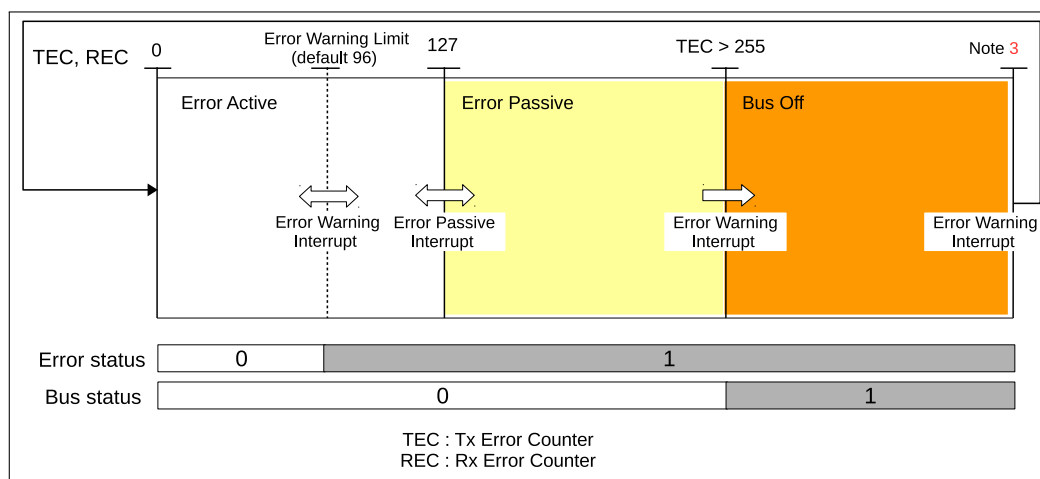


Figure 18-10. Error State Transition

### 18.5.7.1 Error Warning Limit

The Error Warning Limit (EWL) feature is a configurable threshold value for the TEC and REC, where if exceeded, will trigger an interrupt. The EWL is intended to serve as a warning about severe TWAI bus errors, and is triggered before the TWAI controller enters the Error Passive state. The EWL is configured in the [TWAI\\_ERR\\_WARNING\\_LIMIT\\_REG](#) and can only be configured whilst the TWAI controller is in Reset Mode. The [TWAI\\_ERR\\_WARNING\\_LIMIT\\_REG](#) has a default value of 96. When the values of TEC and/or REC are larger than or equal to the EWL value, the [TWAI\\_ERR\\_ST](#) bit is immediately set to 1. Likewise, when the values of both the TEC and REC are smaller than the EWL value, the [TWAI\\_ERR\\_ST](#) bit is immediately reset to 0. The Error Warning Interrupt is triggered whenever the value of the [TWAI\\_ERR\\_ST](#) bit (or the [TWAI\\_BUS\\_OFF\\_ST](#)) changes.

### 18.5.7.2 Error Passive

The TWAI controller is in the Error Passive state when the TEC or REC value exceeds 127. Likewise, when both the TEC and REC are less than or equal to 127, the TWAI controller enters the Error Active state. The Error Passive Interrupt is triggered whenever the TWAI controller transitions from the Error Active state to the Error Passive state or vice versa.

### 18.5.7.3 Bus-Off and Bus-Off Recovery

The TWAI controller enters the Bus-Off state when the TEC value exceeds 255. On entering the Bus-Off state, the TWAI controller will automatically do the following:

- Set REC to 0
- Set TEC to 127
- Set the [TWAI\\_BUS\\_OFF\\_ST](#) bit to 1
- Enter Reset Mode

The Error Warning Interrupt is triggered whenever the value of the [TWAI\\_BUS\\_OFF\\_ST](#) bit (or the [TWAI\\_ERR\\_ST](#) bit) changes.

To return to the Error Active state, the TWAI controller must undergo Bus-Off recovery. Bus-Off recovery requires the TWAI controller to observe 128 occurrences of 11 consecutive Recessive bits on the bus. To initiate Bus-Off recovery (after entering the Bus-Off state), the TWAI controller should enter Operation Mode by setting the [TWAI\\_RESET\\_MODE](#) bit to 0. The TEC tracks the progress of Bus-Off recovery by decrementing the TEC each time the TWAI controller observes 11 consecutive Recessive bits. When Bus-Off recovery has completed (i.e., TEC has decremented from 127 to 0), the [TWAI\\_BUS\\_OFF\\_ST](#) bit will automatically be reset to 0, thus triggering the Error Warning Interrupt.

### 18.5.8 Error Code Capture

The Error Code Capture (ECC) feature allows the TWAI controller to record the error type and bit position of a TWAI bus error in the form of an error code. Upon detecting a TWAI bus error, the Bus Error Interrupt is triggered and the error code is recorded in the [TWAI\\_ERR\\_CODE\\_CAP\\_REG](#). Subsequent bus errors will trigger the Bus Error Interrupt, but their error codes will not be recorded until the current error code is read from the [TWAI\\_ERR\\_CODE\\_CAP\\_REG](#).

The following Table 83 shows the fields of the [TWAI\\_ERR\\_CODE\\_CAP\\_REG](#):

**Table 83: Bit Information of [TWAI\\_ERR\\_CODE\\_CAP\\_REG](#); TWAI Address 0x30**

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	ERRC.1 <sup>1</sup>	ERRC.0 <sup>1</sup>	DIR <sup>2</sup>	SEG.4 <sup>3</sup>	SEG.3 <sup>3</sup>	SEG.2 <sup>3</sup>	SEG.1 <sup>3</sup>	SEG.0 <sup>3</sup>

#### Notes:

- **ERRC:** The Error Code (ERRC) indicates the type of bus error: 00 for bit error, 01 for form error, 10 for stuff error, 11 for other type of error.
- **DIR:** The Direction (DIR) indicates whether the TWAI controller was transmitting or receiving when the bus error: 0 for Transmitter, 1 for Receiver.
- **SEG:** The Error Segment (SEG) indicates which segment of the TWAI message (i.e., bit position) the bus error occurred at.

The following Table 84 shows how to interpret the SEG.0 to SEG.4 bits.

**Table 84: Bit Information of Bits SEG.4 - SEG.0**

Bit SEG.4	Bit SEG.3	Bit SEG.2	Bit SEG.1	Bit SEG.0	Description
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 <sup>1</sup>
0	0	1	0	1	bit IDE <sup>2</sup>
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

Bit SEG.4	Bit SEG.3	Bit SEG.2	Bit SEG.1	Bit SEG.0	Description
0	1	0	0	0	CRC sequence
1	1	0	0	0	CRC delimiter
1	1	0	0	1	acknowledge slot
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

**Notes:**

- Bit RTR: under Standard Frame Format.
- Identifier Extension Bit: 0 for Standard Frame Format.

**18.5.9 Arbitration Lost Capture**

The Arbitration Lost Capture (ALC) feature allows the TWAI controller to record the bit position where it loses arbitration. When the TWAI controller loses arbitration, the bit position is recorded in the [TWAI\\_ARB\\_LOST\\_CAP\\_REG](#) and the Arbitration Lost Interrupt is triggered.

Subsequent losses in arbitration will trigger the Arbitration Lost Interrupt, but will not be recorded in the [TWAI\\_ARB\\_LOST\\_CAP\\_REG](#) until the current Arbitration Lost Capture is read from the [TWAI\\_ERR\\_CODE\\_CAP\\_REG](#).

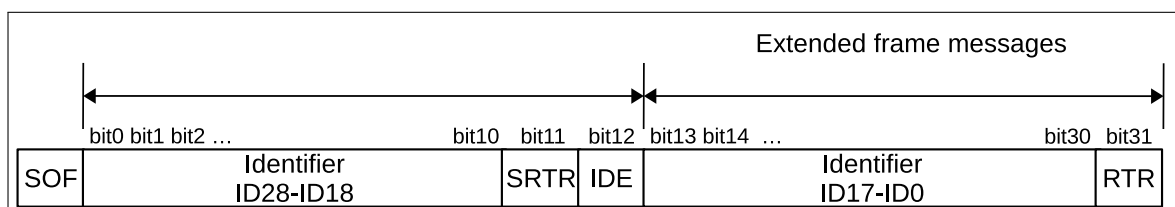
Table 85 illustrates the bit fields of the [TWAI\\_ERR\\_CODE\\_CAP\\_REG](#) whilst Figure 18-11 illustrates the bit positions of a TWAI message.

**Table 85: Bit Information of [TWAI\\_ARB\\_LOST\\_CAP\\_REG](#); TWAI Address 0x2c**

Bit 31-5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	BITNO.4 <sup>1</sup>	BITNO.3 <sup>1</sup>	BITNO.2 <sup>1</sup>	BITNO.1 <sup>1</sup>	BITNO.0 <sup>1</sup>

**Notes:**

- BITNO: Bit Number (BITNO) indicates the nth bit of a TWAI message where arbitration was lost.



**Figure 18-11. Positions of Arbitration Lost Bits**

**18.6 Base Address**

Users can access the TWAI with two base addresses, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter: [1 System and Memory](#).

Table 86: TWAI Base Address

Bus to Access Peripheral	Base Address
PeriBUS1	0x6002B000
PeriBUS2	0x3FC2B000

## 18.7 Register Summary

Name	Description	Address	Access
<b>Configuration Registers</b>			
<a href="#">TWAI_MODE_REG</a>	Mode Register	0x0000	R/W
<a href="#">TWAI_BUS_TIMING_0_REG</a>	Bus Timing Register 0	0x0018	RO   R/W
<a href="#">TWAI_BUS_TIMING_1_REG</a>	Bus Timing Register 1	0x001C	RO   R/W
<a href="#">TWAI_ERR_WARNING_LIMIT_REG</a>	Error Warning Limit Register	0x0034	RO   R/W
<a href="#">TWAI_DATA_0_REG</a>	Data Register 0	0x0040	WO   R/W
<a href="#">TWAI_DATA_1_REG</a>	Data Register 1	0x0044	WO   R/W
<a href="#">TWAI_DATA_2_REG</a>	Data Register 2	0x0048	WO   R/W
<a href="#">TWAI_DATA_3_REG</a>	Data Register 3	0x004C	WO   R/W
<a href="#">TWAI_DATA_4_REG</a>	Data Register 4	0x0050	WO   R/W
<a href="#">TWAI_DATA_5_REG</a>	Data Register 5	0x0054	WO   R/W
<a href="#">TWAI_DATA_6_REG</a>	Data Register 6	0x0058	WO   R/W
<a href="#">TWAI_DATA_7_REG</a>	Data Register 7	0x005C	WO   R/W
<a href="#">TWAI_DATA_8_REG</a>	Data Register 8	0x0060	WO   RO
<a href="#">TWAI_DATA_9_REG</a>	Data Register 9	0x0064	WO   RO
<a href="#">TWAI_DATA_10_REG</a>	Data Register 10	0x0068	WO   RO
<a href="#">TWAI_DATA_11_REG</a>	Data Register 11	0x006C	WO   RO
<a href="#">TWAI_DATA_12_REG</a>	Data Register 12	0x0070	WO   RO
<a href="#">TWAI_CLOCK_DIVIDER_REG</a>	Clock Divider Register	0x007C	varies
<b>Control Registers</b>			
<a href="#">TWAI_CMD_REG</a>	Command Register	0x0004	WO
<b>Status Registers</b>			
<a href="#">TWAI_STATUS_REG</a>	Status Register	0x0008	RO
<a href="#">TWAI_ARB_LOST_CAP_REG</a>	Arbitration Lost Capture Register	0x002C	RO
<a href="#">TWAI_ERR_CODE_CAP_REG</a>	Error Code Capture Register	0x0030	RO
<a href="#">TWAI_RX_ERR_CNT_REG</a>	Receive Error Counter Register	0x0038	RO   R/W
<a href="#">TWAI_TX_ERR_CNT_REG</a>	Transmit Error Counter Register	0x003C	RO   R/W
<a href="#">TWAI_RX_MESSAGE_CNT_REG</a>	Receive Message Counter Register	0x0074	RO
<b>Interrupt Registers</b>			
<a href="#">TWAI_INT_RAW_REG</a>	Interrupt Register	0x000C	RO
<a href="#">TWAI_INT_ENA_REG</a>	Interrupt Enable Register	0x0010	R/W

## 18.8 Register Description

Register 18.1: TWAI\_MODE\_REG (0x0000)

(reserved)																												TWAI_RX_FILTER_MODE TWAI_SELF_TEST_MODE TWAI_LISTEN_ONLY_MODE TWAI_RESET_MODE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
31																												4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

**TWAI\_RESET\_MODE** This bit is used to configure the operating mode of the TWAI Controller. 1: Reset mode; 0: Operating mode (R/W)

**TWAI\_LISTEN\_ONLY\_MODE** 1: Listen only mode. In this mode the nodes will only receive messages from the bus, without generating the acknowledge signal nor updating the RX error counter. (R/W)

**TWAI\_SELF\_TEST\_MODE** 1: Self test mode. In this mode the TX nodes can perform a successful transmission without receiving the acknowledge signal. This mode is often used to test a single node with the self reception request command. (R/W)

**TWAI\_RX\_FILTER\_MODE** This bit is used to configure the filter mode. 0: Dual filter mode; 1: Single filter mode (R/W)

Register 18.2: TWAI\_BUS\_TIMING\_0\_REG (0x0018)

(reserved)																TWAI_SYNC_JUMP_WIDTH				TWAI_BAUD_PRESC				
31													16	15	14	13								0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0	0x00							Reset

**TWAI\_BAUD\_PRESC** Baud Rate Prescaler, determines the frequency dividing ratio. (RO | R/W)

**TWAI\_SYNC\_JUMP\_WIDTH** Synchronization Jump Width (SJW), 1 ~ 14 T<sub>q</sub> wide. (RO | R/W)

Register 18.3: TWAI\_BUS\_TIMING\_1\_REG (0x001C)

(reserved)																								TWAI_TIME_SAMP		TWAI_TIME_SEG2		TWAI_TIME_SEG1		
31																								8	7	6	4	3	0	
0 0																								0	0	0x0		0x0		Reset

**TWAI\_TIME\_SEG1** The width of PBS1. (RO | R/W)

**TWAI\_TIME\_SEG2** The width of PBS2. (RO | R/W)

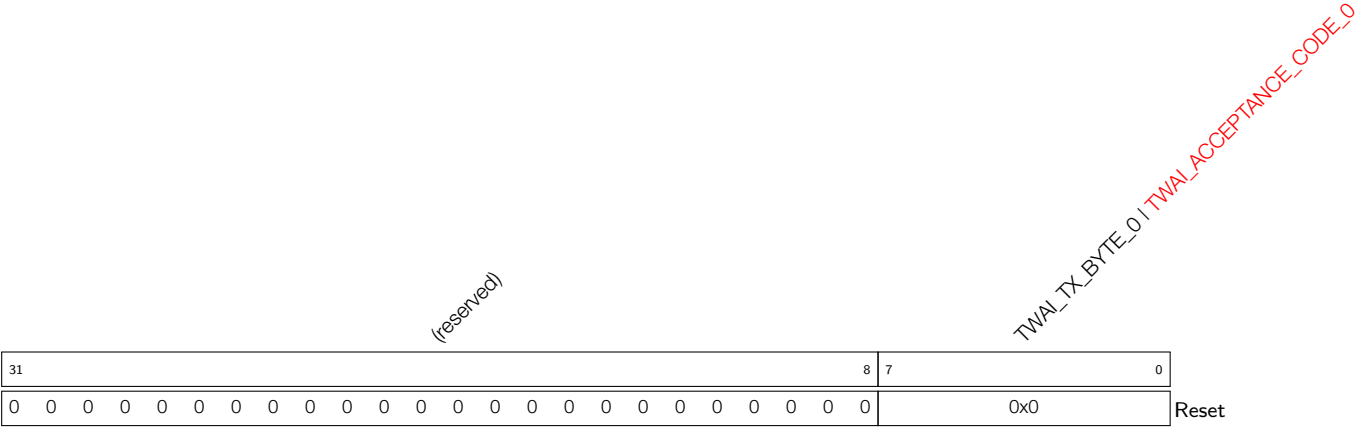
**TWAI\_TIME\_SAMP** The number of sample points. 0: the bus is sampled once; 1: the bus is sampled three times (RO | R/W)

Register 18.4: TWAI\_ERR\_WARNING\_LIMIT\_REG (0x0034)

(reserved)																												TWAI_ERR_WARNING_LIMIT																											
31																												7														0													
0 0																												0x60														Reset													

**TWAI\_ERR\_WARNING\_LIMIT** Error warning threshold. In the case when any of a error counter value exceeds the threshold, or all the error counter values are below the threshold, an error warning interrupt will be triggered (given the enable signal is valid). (RO | R/W)

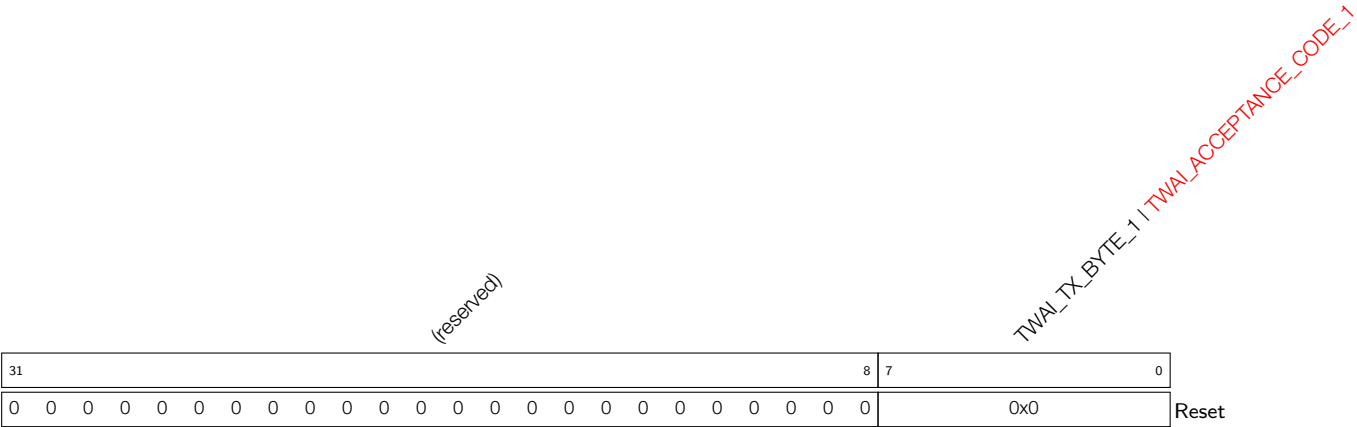
Register 18.5: TWAI\_DATA\_0\_REG (0x0040)



**TWAI\_TX\_BYTE\_0** Stored the 0th byte information of the data to be transmitted under operating mode. (WO)

**TWAI\_ACCEPTANCE\_CODE\_0** Stored the 0th byte of the filter code under reset mode. (R/W)

Register 18.6: TWAI\_DATA\_1\_REG (0x0044)

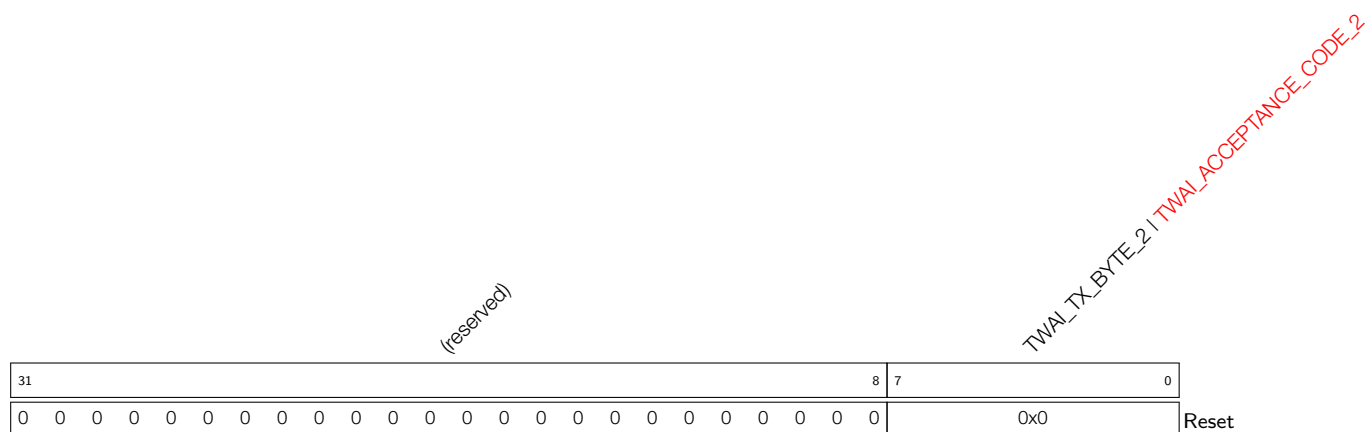


**TWAI\_TX\_BYTE\_1** Stored the 1st byte information of the data to be transmitted under operating mode. (WO)

**TWAI\_ACCEPTANCE\_CODE\_1** Stored the 1st byte of the filter code under reset mode. (R/W)



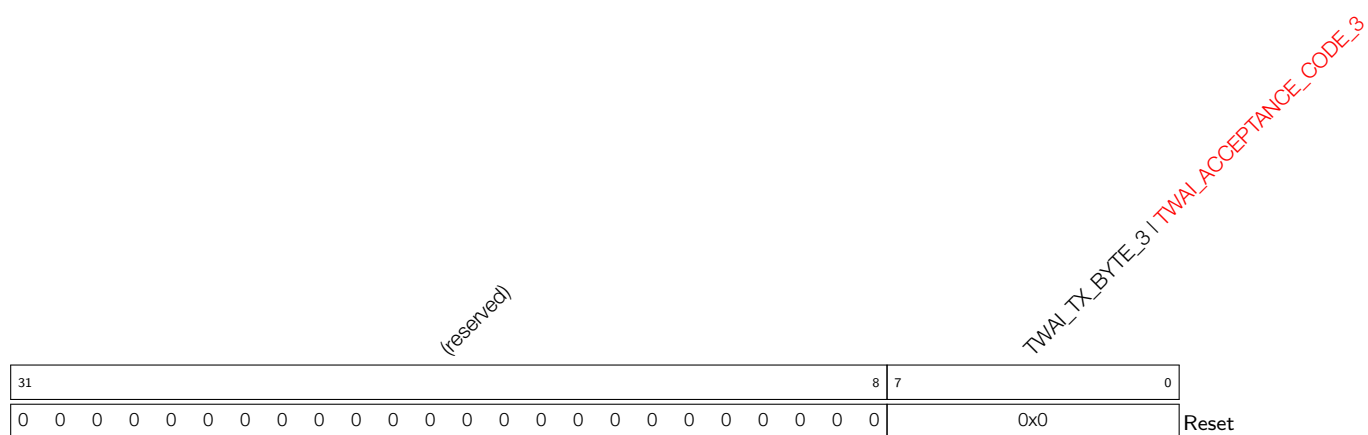
## Register 18.7: TWAI\_DATA\_2\_REG (0x0048)



**TWAI\_TX\_BYTE\_2** Stored the 2nd byte information of the data to be transmitted under operating mode. (WO)

**TWAI\_ACCEPTANCE\_CODE\_2** Stored the 2nd byte of the filter code under reset mode. (R/W)

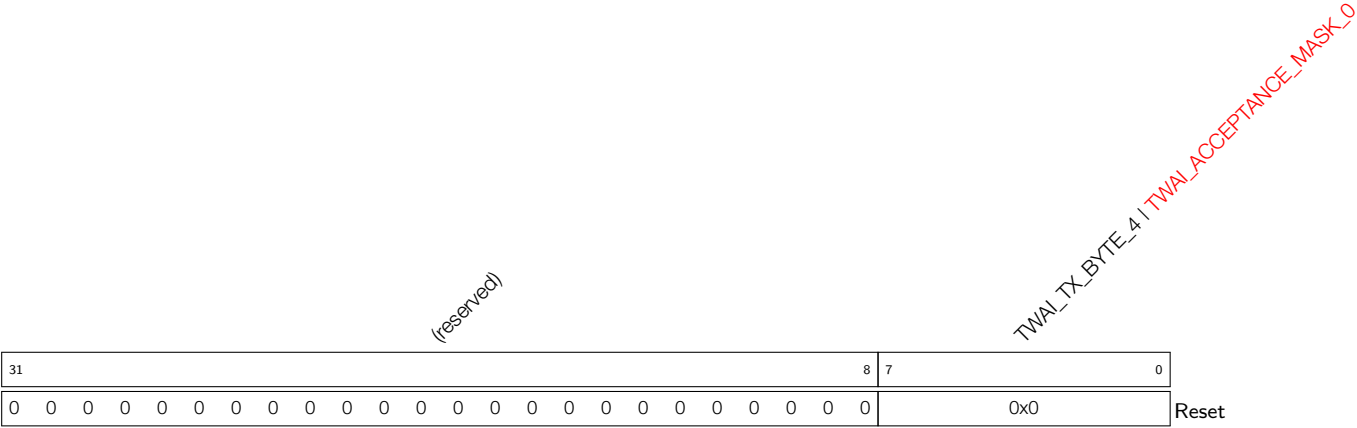
## Register 18.8: TWAI\_DATA\_3\_REG (0x004C)



**TWAI\_TX\_BYTE\_3** Stored the 3rd byte information of the data to be transmitted under operating mode. (WO)

**TWAI\_ACCEPTANCE\_CODE\_3** Stored the 3rd byte of the filter code under reset mode. (R/W)

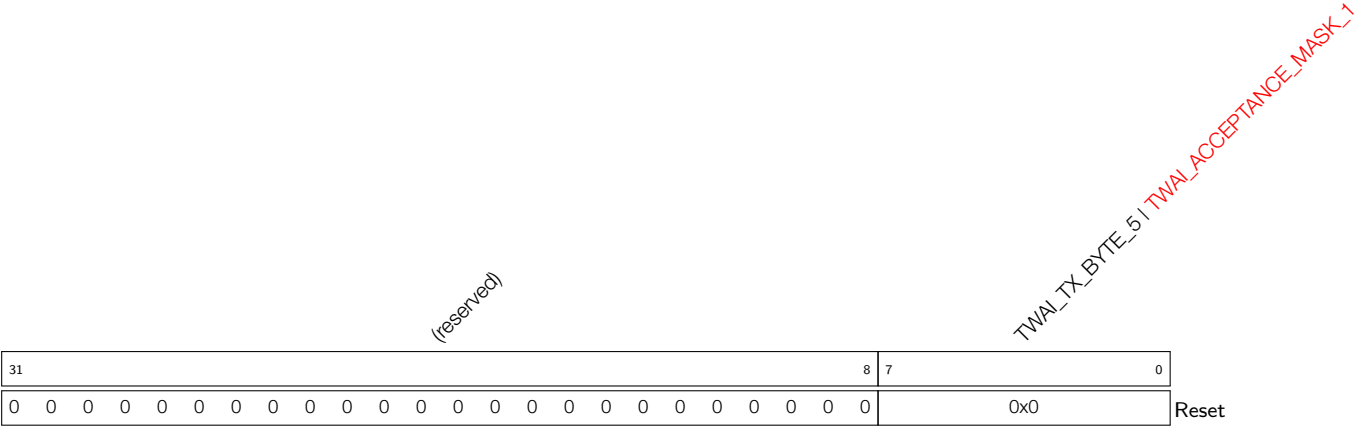
Register 18.9: TWAI\_DATA\_4\_REG (0x0050)



**TWAI\_TX\_BYTE\_4** Stored the 4th byte information of the data to be transmitted under operating mode. (WO)

**TWAI\_ACCEPTANCE\_MASK\_0** Stored the 0th byte of the filter code under reset mode. (R/W)

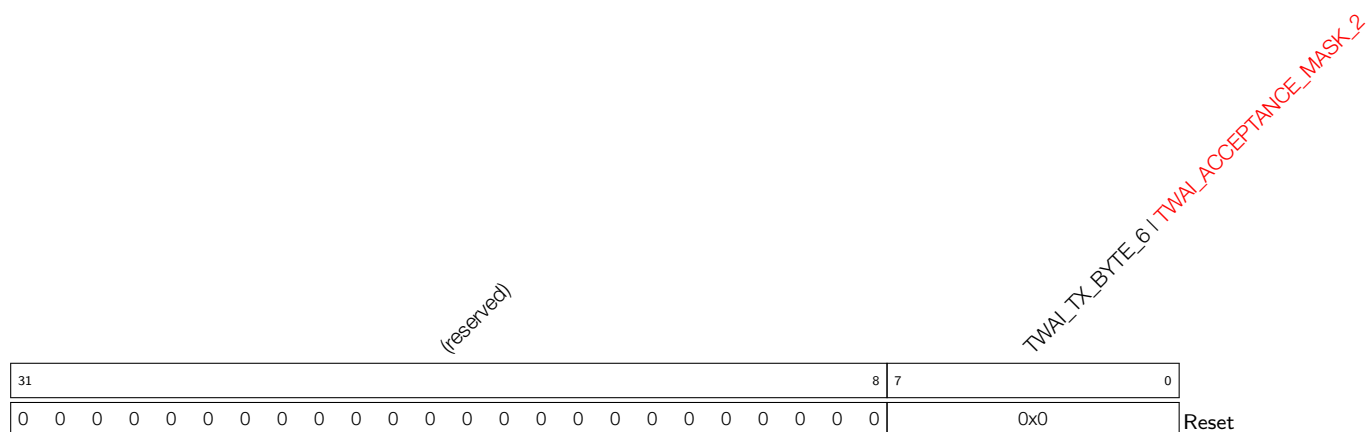
Register 18.10: TWAI\_DATA\_5\_REG (0x0054)



**TWAI\_TX\_BYTE\_5** Stored the 5th byte information of the data to be transmitted under operating mode. (WO)

**TWAI\_ACCEPTANCE\_MASK\_1** Stored the 1st byte of the filter code under reset mode. (R/W)

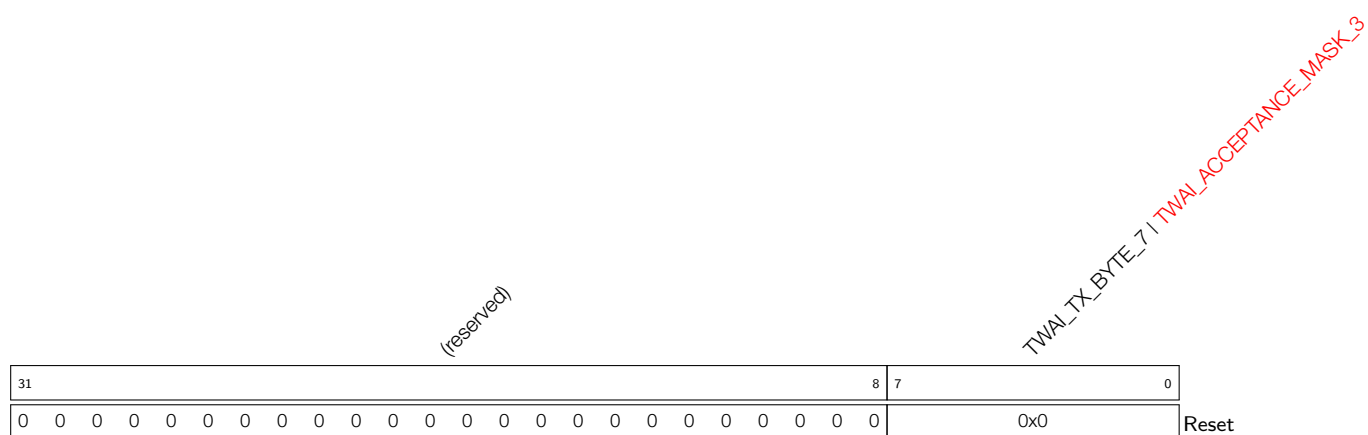
## Register 18.11: TWAI\_DATA\_6\_REG (0x0058)



**TWAI\_TX\_BYTE\_6** Stored the 6th byte information of the data to be transmitted under operating mode. (WO)

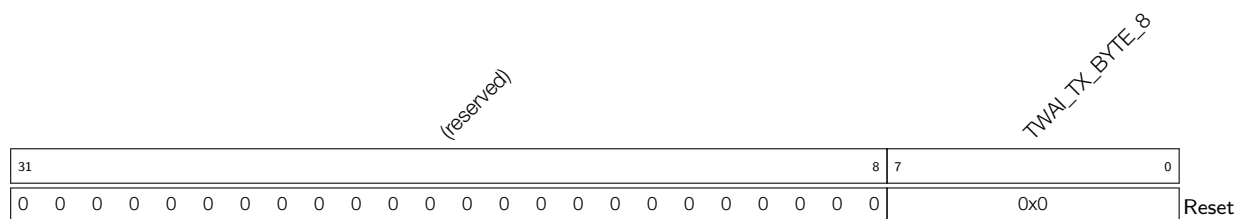
**TWAI\_ACCEPTANCE\_MASK\_2** Stored the 2nd byte of the filter code under reset mode. (R/W)

## Register 18.12: TWAI\_DATA\_7\_REG (0x005C)

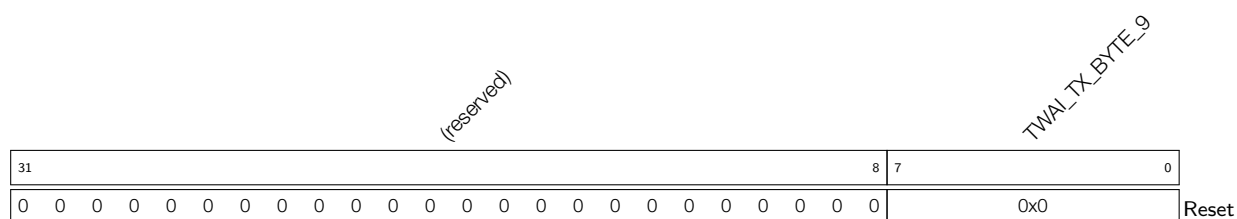


**TWAI\_TX\_BYTE\_7** Stored the 7th byte information of the data to be transmitted under operating mode. (WO)

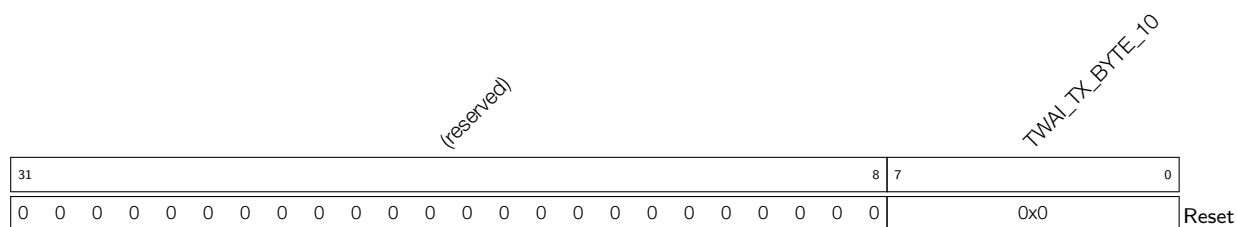
**TWAI\_ACCEPTANCE\_MASK\_3** Stored the 3rd byte of the filter code under reset mode. (R/W)

**Register 18.13: TWAI\_DATA\_8\_REG (0x0060)**

**TWAI\_TX\_BYTE\_8** Stored the 8th byte information of the data to be transmitted under operating mode. (WO)

**Register 18.14: TWAI\_DATA\_9\_REG (0x0064)**

**TWAI\_TX\_BYTE\_9** Stored the 9th byte information of the data to be transmitted under operating mode. (WO)

**Register 18.15: TWAI\_DATA\_10\_REG (0x0068)**

**TWAI\_TX\_BYTE\_10** Stored the 10th byte information of the data to be transmitted under operating mode. (WO)

**Register 18.16: TWAI\_DATA\_11\_REG (0x006C)**

(reserved)																								TWAI_TX_BYTE_11									
31																								8	7	0							
0 0																								0x0								Reset	

**TWAI\_TX\_BYTE\_11** Stored the 11th byte information of the data to be transmitted under operating mode. (WO)

**Register 18.17: TWAI\_DATA\_12\_REG (0x0070)**

(reserved)																TWAI_TX_BYTE_12																	
31																8	7	0															
0 0																0x0																Reset	

**TWAI\_TX\_BYTE\_12** Stored the 12th byte information of the data to be transmitted under operating mode. (WO)

**Register 18.18: TWAI\_CLOCK\_DIVIDER\_REG (0x007C)**

(reserved)																								TWAI_CLOCK_OFF		TWAI_CD																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
31																			9	8	7												0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**TWAI\_CD** These bits are used to configure frequency dividing coefficients of the external CLKOUT pin. (R/W)

**TWAI\_CLOCK\_OFF** This bit can be configured under reset mode. 1: Disable the external CLKOUT pin; 0: Enable the external CLKOUT pin (RO | R/W)

**Register 18.19: TWAI\_CMD\_REG (0x0004)**

(reserved)																								TWAI_SELF_RX_REQ TWAI_CLR_OVERRUN TWAI_RELEASE_BUF TWAI_ABORT_TX TWAI_TX_REQ																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
31																								5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**TWAI\_TX\_REQ** Set the bit to 1 to allow the driving nodes start transmission. (WO)

**TWAI\_ABORT\_TX** Set the bit to 1 to cancel a pending transmission request. (WO)

**TWAI\_RELEASE\_BUF** Set the bit to 1 to release the RX buffer. (WO)

**TWAI\_CLR\_OVERRUN** Set the bit to 1 to clear the data overrun status bit. (WO)

**TWAI\_SELF\_RX\_REQ** Self reception request command. Set the bit to 1 to allow a message be transmitted and received simultaneously. (WO)

**Register 18.20: TWAI\_STATUS\_REG (0x0008)**

(reserved)																						TWAI_MISS_ST TWAI_BUS_OFF_ST TWAI_ERR_ST TWAI_TX_ST TWAI_RX_ST TWAI_TX_COMPLETE TWAI_TX_BUF_ST TWAI_OVERRUN_ST TWAI_RX_BUF_ST											
31																						9	8	7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	Reset			

**TWAI\_RX\_BUF\_ST** 1: The data in the RX buffer is not empty, with at least one received data packet. (RO)

**TWAI\_OVERRUN\_ST** 1: The RX FIFO is full and data overrun has occurred. (RO)

**TWAI\_TX\_BUF\_ST** 1: The TX buffer is empty, the CPU may write a message into it. (RO)

**TWAI\_TX\_COMPLETE** 1: The TWAI controller has successfully received a packet from the bus. (RO)

**TWAI\_RX\_ST** 1: The TWAI Controller is receiving a message from the bus. (RO)

**TWAI\_TX\_ST** 1: The TWAI Controller is transmitting a message to the bus. (RO)

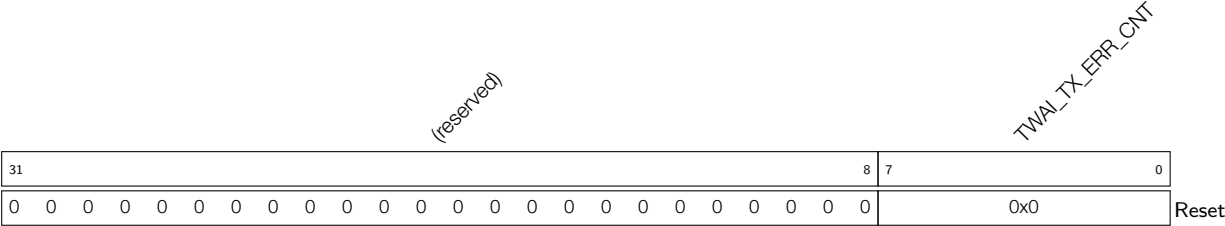
**TWAI\_ERR\_ST** 1: At least one of the RX/TX error counter has reached or exceeded the value set in register [TWAI\\_ERR\\_WARNING\\_LIMIT\\_REG](#). (RO)

**TWAI\_BUS\_OFF\_ST** 1: In bus-off status, the TWAI Controller is no longer involved in bus activities. (RO)

**TWAI\_MISS\_ST** This bit reflects whether the data packet in the RX FIFO is complete. 1: The current packet is missing; 0: The current packet is complete (RO)

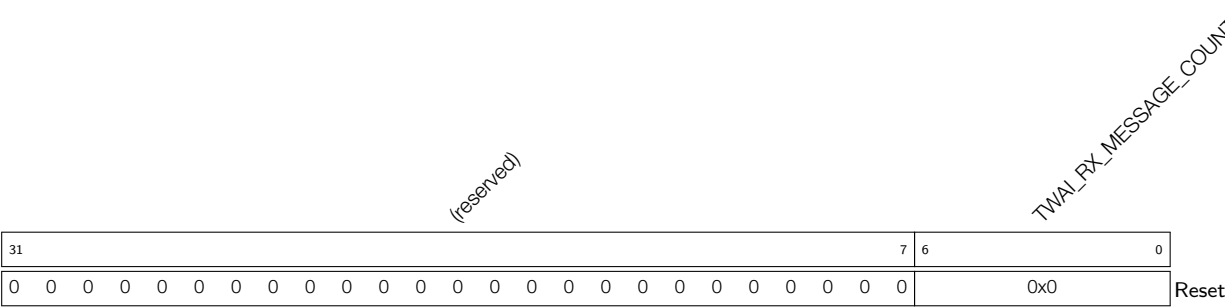


Register 18.24: TWAI\_TX\_ERR\_CNT\_REG (0x003C)



**TWAI\_TX\_ERR\_CNT** The TX error counter register, reflects value changes under transmission status.  
(RO | R/W)

Register 18.25: TWAI\_RX\_MESSAGE\_CNT\_REG (0x0074)



**TWAI\_RX\_MESSAGE\_COUNTER** This register reflects the number of messages available within the RX FIFO. (RO)



Register 18.26: TWAI\_INT\_RAW\_REG (0x000C)

(reserved)																TWAI_BUS_ERR_INT_ST			
																TWAI_ARB_LOST_INT_ST			
																TWAI_ERR_PASSIVE_INT_ST			
																(reserved)			
																TWAI_OVERRUN_INT_ST			
																TWAI_ERR_WARN_INT_ST			
																TWAI_TX_INT_ST			
																TWAI_RX_INT_ST			
31								8	7	6	5	4	3	2	1	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**TWAI\_RX\_INT\_ST** Receive interrupt. If this bit is set to 1, it indicates there are messages to be handled in the RX FIFO. (RO)

**TWAI\_TX\_INT\_ST** Transmit interrupt. If this bit is set to 1, it indicates the message transmitting mission is finished and a new transmission is able to execute. (RO)

**TWAI\_ERR\_WARN\_INT\_ST** Error warning interrupt. If this bit is set to 1, it indicates the error status signal and the bus-off status signal of Status register have changed (e.g., switched from 0 to 1 or from 1 to 0). (RO)

**TWAI\_OVERRUN\_INT\_ST** Data overrun interrupt. If this bit is set to 1, it indicates a data overrun interrupt is generated in the RX FIFO. (RO)

**TWAI\_ERR\_PASSIVE\_INT\_ST** Error passive interrupt. If this bit is set to 1, it indicates the TWAI Controller is switched between error active status and error passive status due to the change of error counters. (RO)

**TWAI\_ARB\_LOST\_INT\_ST** Arbitration lost interrupt. If this bit is set to 1, it indicates an arbitration lost interrupt is generated. (RO)

**TWAI\_BUS\_ERR\_INT\_ST** Error interrupt. If this bit is set to 1, it indicates an error is detected on the bus. (RO)

(reserved)

TWAL\_BUS\_ERR\_INT\_ENA  
TWAL\_ARB\_LOST\_INT\_ENA  
TWAL\_ERR\_PASSIVE\_INT\_ENA  
(reserved)  
TWAL\_OVERRUN\_INT\_ENA  
TWAL\_ERR\_WARN\_INT\_ENA  
TWAL\_TX\_INT\_ENA  
TWAL\_RX\_INT\_ENA

**TWAI\_TX\_INT\_ENA** Set this bit to 1 to enable transmit interrupt. (R/W)

**TWAI\_OVERRUN\_INT\_ENA** Set this bit to 1 to enable data overrun interrupt. (R/W)

**TWAI\_ARB\_LOST\_INT\_ENA** Set this bit to 1 to enable arbitration lost interrupt. (R/W)

**TWAI\_BUS\_ERR\_INT\_ENA** Set this bit to 1 to enable error interrupt. (R/W)

## 19. AES Accelerator

### 19.1 Introduction

ESP32-S2 integrates an Advanced Encryption Standard (AES) Accelerator, which is a hardware device that speeds up AES Algorithm significantly, compared to AES algorithms implemented solely in software. The AES Accelerator integrated in ESP32-S2 has two working modes, which are [Typical AES](#) and [DMA-AES](#).

### 19.2 Features

The following functionality is supported:

- Typical AES working mode
  - AES-128/AES-192/AES-256 encryption and decryption
  - Four variations of key endianness and four variations of text endianness
- DMA-AES working mode
  - Block mode
    - \* ECB (Electronic Codebook)
    - \* CBC (Cipher Block Chaining)
    - \* OFB (Output Feedback)
    - \* CTR (Counter)
    - \* CFB8 (8-bit Cipher Feedback)
    - \* CFB128 (128-bit Cipher Feedback)
  - GCM (Galois/Counter Mode)
  - Interrupt on completion of computation

### 19.3 Working Modes

The AES Accelerator integrated in ESP32-S2 has two working modes, which are [Typical AES](#) and [DMA-AES](#).

- Typical AES Working Mode: supports AES-128/AES-192/AES-256 encryption and decryption under [NIST FIPS 197](#). In this working mode, the plaintext and ciphertext is written and read via CPU directly.
- DMA-AES Working Mode: supports block cipher algorithms ECB/CBC/OFB/CTR/CFB8/CFB128 under [NIST SP 800-38A](#), and GCM mode of operation under [NIST SP 800-38D](#). In this working mode, the plaintext and ciphertext is written and read via crypto DMA. An interrupt will be generated when operation completes.

Users can choose the working mode for AES accelerator by configuring the [AES\\_DMA\\_ENABLE\\_REG](#) register according to Table [88](#) below.

**Table 88: AES Accelerator Working Mode**

AES_DMA_ENABLE_REG	Working Mode
0	Typical AES
1	DMA-AES

For detailed introduction on these two working modes, please refer to Section 19.4 and Section 19.5 below.

**Notice:**

ESP32-S2's [Digital Signature](#) and [External Memory Manual Encryption](#) modules also call the AES accelerator. Therefore, users cannot access the AES accelerator when these modules are working.

## 19.4 Typical AES Working Mode

In the Typical AES working mode, the AES accelerator is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data, i.e. AES-128/AES-192/AES-256 encryption and decryption. Users can choose the operation type for AES accelerator working in Typical AES working mode by configuring the [AES\\_MODE\\_REG](#) register according to Table 89 below.

**Table 89: Operation Type under Typical AES Working Mode**

AES_MODE_REG[2:0]	Operation Type
0	AES-128 encryption
1	AES-192 encryption
2	AES-256 encryption
4	AES-128 decryption
5	AES-192 decryption
6	AES-256 decryption

Users can check the working status of the AES accelerator by inquiring the [AES\\_STATE\\_REG](#) register and comparing the return value against the Table 90 below.

**Table 90: Working Status under Typical AES Working Mode**

AES_STATE_REG	Status	Description
0	IDLE	The AES accelerator is idle or completed operation.
1	WORK	The AES accelerator is in the middle of an operation.

In the Typical AES working mode, the AES accelerator requires 11 ~ 15 clock cycles to encrypt a message block, and 21 or 22 clock cycles to decrypt a message block.

### 19.4.1 Key, Plaintext, and Ciphertext

The encryption or decryption key is stored in [AES\\_KEY\\_n\\_REG](#), which is a set of eight 32-bit registers.

- For AES-128 encryption/decryption, the 128-bit key is stored in [AES\\_KEY\\_0\\_REG](#) ~ [AES\\_KEY\\_3\\_REG](#).

- For AES-192 encryption/decryption, the 192-bit key is stored in [AES\\_KEY\\_0\\_REG ~ AES\\_KEY\\_5\\_REG](#).
- For AES-256 encryption/decryption, the 256-bit key is stored in [AES\\_KEY\\_0\\_REG ~ AES\\_KEY\\_7\\_REG](#).

The plaintext and ciphertext are stored in [AES\\_TEXT\\_IN\\_m\\_REG](#) and [AES\\_TEXT\\_OUT\\_m\\_REG](#), which are two sets of four 32-bit registers.

- For AES-128/AES-192/AES-256 encryption, the [AES\\_TEXT\\_IN\\_m\\_REG](#) registers are initialized with plaintext. Then, the AES Accelerator stores the ciphertext into [AES\\_TEXT\\_OUT\\_m\\_REG](#) after operation.
- For AES-128/AES-192/AES-256 decryption, the [AES\\_TEXT\\_IN\\_m\\_REG](#) registers are initialized with ciphertext. Then, the AES Accelerator stores the plaintext into [AES\\_TEXT\\_OUT\\_m\\_REG](#) after operation.

## 19.4.2 Endianness

### Text Endianness

In Typical AES working mode, the AES Accelerator uses cryptographic keys to encrypt and decrypt data in blocks of 128 bits. The Bit 2 and Bit 3 of the [AES\\_ENDIAN\\_REG](#) register define the endianness of input text, while the Bit 4 and Bit 5 define the endianness of output text. To be more specific, Bit 2 and Bit 4 control how the four bytes are stored in each word, and Bit 3 and Bit 5 control how the four words are stored in each message block.

Users can choose one of the four text endianness types provided by the AES Accelerator by configuring the [AES\\_ENDIAN\\_REG](#) register. Details can be seen in Table 91.

**Table 91: Text Endianness Types for Typical AES**

Word Endian Controlling Bit	Byte Endian Controlling Bit	Plaintext/Ciphertext <sup>2</sup>					
0	0	State <sup>1</sup>		c			
				0	1	2	3
		r	0	AES_TEXT_x_3_REG[31:24]	AES_TEXT_x_2_REG[31:24]	AES_TEXT_x_1_REG[31:24]	AES_TEXT_x_0_REG[31:24]
			1	AES_TEXT_x_3_REG[23:16]	AES_TEXT_x_2_REG[23:16]	AES_TEXT_x_1_REG[23:16]	AES_TEXT_x_0_REG[23:16]
			2	AES_TEXT_x_3_REG[15:8]	AES_TEXT_x_2_REG[15:8]	AES_TEXT_x_1_REG[15:8]	AES_TEXT_x_0_REG[15:8]
3	AES_TEXT_x_3_REG[7:0]	AES_TEXT_x_2_REG[7:0]	AES_TEXT_x_1_REG[7:0]	AES_TEXT_x_0_REG[7:0]			
0	1	State		c			
				0	1	2	3
		r	0	AES_TEXT_x_3_REG[7:0]	AES_TEXT_x_2_REG[7:0]	AES_TEXT_x_1_REG[7:0]	AES_TEXT_x_0_REG[7:0]
			1	AES_TEXT_x_3_REG[15:8]	AES_TEXT_x_2_REG[15:8]	AES_TEXT_x_1_REG[15:8]	AES_TEXT_x_0_REG[15:8]
			2	AES_TEXT_x_3_REG[23:16]	AES_TEXT_x_2_REG[23:16]	AES_TEXT_x_1_REG[23:16]	AES_TEXT_x_0_REG[23:16]
3	AES_TEXT_x_3_REG[31:24]	AES_TEXT_x_2_REG[31:24]	AES_TEXT_x_1_REG[31:24]	AES_TEXT_x_0_REG[31:24]			
1	0	State		c			
				0	1	2	3
		r	0	AES_TEXT_x_0_REG[31:24]	AES_TEXT_x_1_REG[31:24]	AES_TEXT_x_2_REG[31:24]	AES_TEXT_x_3_REG[31:24]
			1	AES_TEXT_x_0_REG[23:16]	AES_TEXT_x_1_REG[23:16]	AES_TEXT_x_2_REG[23:16]	AES_TEXT_x_3_REG[23:16]
			2	AES_TEXT_x_0_REG[15:8]	AES_TEXT_x_1_REG[15:8]	AES_TEXT_x_2_REG[15:8]	AES_TEXT_x_3_REG[15:8]
3	AES_TEXT_x_0_REG[7:0]	AES_TEXT_x_1_REG[7:0]	AES_TEXT_x_2_REG[7:0]	AES_TEXT_x_3_REG[7:0]			
1	1	State		c			
				0	1	2	3
		r	0	AES_TEXT_x_0_REG[7:0]	AES_TEXT_x_1_REG[7:0]	AES_TEXT_x_2_REG[7:0]	AES_TEXT_x_3_REG[7:0]
			1	AES_TEXT_x_0_REG[15:8]	AES_TEXT_x_1_REG[15:8]	AES_TEXT_x_2_REG[15:8]	AES_TEXT_x_3_REG[15:8]
			2	AES_TEXT_x_0_REG[23:16]	AES_TEXT_x_1_REG[23:16]	AES_TEXT_x_2_REG[23:16]	AES_TEXT_x_3_REG[23:16]
3	AES_TEXT_x_0_REG[31:24]	AES_TEXT_x_1_REG[31:24]	AES_TEXT_x_2_REG[31:24]	AES_TEXT_x_3_REG[31:24]			

**Note:**

1. The definition of “State” is described in Section 3.4 The State in [NIST FIPS 197](#).
2. Where,
  - When  $x$  = IN, the Word Endian and Byte Endian controlling bits of [AES\\_TEXT\\_IN\\_](#) $m$ \_REG are the Bit 2 and Bit 3 of [AES\\_ENDIAN\\_REG](#), respectively;
  - When  $x$  = OUT, the Word Endian and Byte Endian controlling bits of [AES\\_TEXT\\_OUT\\_](#) $m$ \_REG are the Bit 4 and Bit 5 of [AES\\_ENDIAN\\_REG](#), respectively.

**Key Endianness**

In Typical AES working mode, Bit 0 and bit 1 in [AES\\_ENDIAN\\_REG](#) define the key endianness.

Users can choose one of the four key endianness types provided by the AES accelerator by configuring the [AES\\_ENDIAN\\_REG](#) register. Details can be seen in Table 92, Table 93, and Table 94.

Table 92: Key Endianness Types for AES-128 Encryption and Decryption

AES_ENDIAN_REG[1]	AES_ENDIAN_REG[0]	Bit <sup>2</sup>	w[0]	w[1]	w[2]	w[3] <sup>1</sup>
0	0	[31:24]	AES_KEY_3_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_0_REG[31:24]
		[23:16]	AES_KEY_3_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_0_REG[23:16]
		[15:8]	AES_KEY_3_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_0_REG[15:8]
		[7:0]	AES_KEY_3_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_0_REG[7:0]
0	1	[31:24]	AES_KEY_3_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_0_REG[7:0]
		[23:16]	AES_KEY_3_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_0_REG[15:8]
		[15:8]	AES_KEY_3_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_0_REG[23:16]
		[7:0]	AES_KEY_3_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_0_REG[31:24]
1	0	[31:24]	AES_KEY_0_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_3_REG[31:24]
		[23:16]	AES_KEY_0_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_3_REG[23:16]
		[15:8]	AES_KEY_0_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_3_REG[15:8]
		[7:0]	AES_KEY_0_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_3_REG[7:0]
1	1	[31:24]	AES_KEY_0_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_3_REG[7:0]
		[23:16]	AES_KEY_0_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_3_REG[15:8]
		[15:8]	AES_KEY_0_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_3_REG[23:16]
		[7:0]	AES_KEY_0_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_3_REG[31:24]

Note:

1. w[0] ~ w[3] are “the first Nk words of the expanded key” as specified in Section 5.2 Key Expansion in [NIST FIPS 197](#).
2. “Column Bit” specifies the bytes of each word stored in w[0] ~ w[3].

Table 93: Key Endianness Types for AES-192 Encryption and Decryption

AES_ENDIAN_REG[1]	AES_ENDIAN_REG[0]	Bit <sup>2</sup>	w[0]	w[1]	w[2]	w[3]	w[4]	w[5] <sup>1</sup>
0	0	[31:24]	AES_KEY_5_REG[31:24]	AES_KEY_4_REG[31:24]	AES_KEY_3_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_0_REG[31:24]
		[23:16]	AES_KEY_5_REG[23:16]	AES_KEY_4_REG[23:16]	AES_KEY_3_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_0_REG[23:16]
		[15:8]	AES_KEY_5_REG[15:8]	AES_KEY_4_REG[15:8]	AES_KEY_3_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_0_REG[15:8]
		[7:0]	AES_KEY_5_REG[7:0]	AES_KEY_4_REG[7:0]	AES_KEY_3_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_0_REG[7:0]
0	1	[31:24]	AES_KEY_5_REG[7:0]	AES_KEY_4_REG[7:0]	AES_KEY_3_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_0_REG[7:0]
		[23:16]	AES_KEY_5_REG[15:8]	AES_KEY_4_REG[15:8]	AES_KEY_3_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_0_REG[15:8]
		[15:8]	AES_KEY_5_REG[23:16]	AES_KEY_4_REG[23:16]	AES_KEY_3_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_0_REG[23:16]
		[7:0]	AES_KEY_5_REG[31:24]	AES_KEY_4_REG[31:24]	AES_KEY_3_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_0_REG[31:24]
1	0	[31:24]	AES_KEY_0_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_3_REG[31:24]	AES_KEY_4_REG[31:24]	AES_KEY_5_REG[31:24]
		[23:16]	AES_KEY_0_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_3_REG[23:16]	AES_KEY_4_REG[23:16]	AES_KEY_5_REG[23:16]
		[15:8]	AES_KEY_0_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_3_REG[15:8]	AES_KEY_4_REG[15:8]	AES_KEY_5_REG[15:8]
		[7:0]	AES_KEY_0_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_3_REG[7:0]	AES_KEY_4_REG[7:0]	AES_KEY_5_REG[7:0]
1	1	[31:24]	AES_KEY_0_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_3_REG[7:0]	AES_KEY_4_REG[7:0]	AES_KEY_5_REG[7:0]
		[23:16]	AES_KEY_0_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_3_REG[15:8]	AES_KEY_4_REG[15:8]	AES_KEY_5_REG[15:8]
		[15:8]	AES_KEY_0_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_3_REG[23:16]	AES_KEY_4_REG[23:16]	AES_KEY_5_REG[23:16]
		[7:0]	AES_KEY_0_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_3_REG[31:24]	AES_KEY_4_REG[31:24]	AES_KEY_5_REG[31:24]

Note:

1. w[0] ~ w[5] are “the first Nk words of the expanded key” as specified in Chapter 5.2 Key Expansion in [NIST FIPS 197](#).
2. “Column Bit” specifies the bytes of each word stored in w[0] ~ w[5].

Table 94: Key Endianness Types for AES-256 Encryption and Decryption

AES_ENDIAN_REG[1]	AES_ENDIAN_REG[0]	Bit <sup>2</sup>	w[0]	w[1]	w[2]	w[3]	w[4]	w[5]	w[6]	w[7] <sup>1</sup>
0	0	[31:24]	AES_KEY_7_REG[31:24]	AES_KEY_6_REG[31:24]	AES_KEY_5_REG[31:24]	AES_KEY_4_REG[31:24]	AES_KEY_3_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_0_REG[31:24]
		[23:16]	AES_KEY_7_REG[23:16]	AES_KEY_6_REG[23:16]	AES_KEY_5_REG[23:16]	AES_KEY_4_REG[23:16]	AES_KEY_3_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_0_REG[23:16]
		[15:8]	AES_KEY_7_REG[15:8]	AES_KEY_6_REG[15:8]	AES_KEY_5_REG[15:8]	AES_KEY_4_REG[15:8]	AES_KEY_3_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_0_REG[15:8]
		[7:0]	AES_KEY_7_REG[7:0]	AES_KEY_6_REG[7:0]	AES_KEY_5_REG[7:0]	AES_KEY_4_REG[7:0]	AES_KEY_3_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_0_REG[7:0]
0	1	[31:24]	AES_KEY_7_REG[7:0]	AES_KEY_6_REG[7:0]	AES_KEY_5_REG[7:0]	AES_KEY_4_REG[7:0]	AES_KEY_3_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_0_REG[7:0]
		[23:16]	AES_KEY_7_REG[15:8]	AES_KEY_6_REG[15:8]	AES_KEY_5_REG[15:8]	AES_KEY_4_REG[15:8]	AES_KEY_3_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_0_REG[15:8]
		[15:8]	AES_KEY_7_REG[23:16]	AES_KEY_6_REG[23:16]	AES_KEY_5_REG[23:16]	AES_KEY_4_REG[23:16]	AES_KEY_3_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_0_REG[23:16]
		[7:0]	AES_KEY_7_REG[31:24]	AES_KEY_6_REG[31:24]	AES_KEY_5_REG[31:24]	AES_KEY_4_REG[31:24]	AES_KEY_3_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_0_REG[31:24]
1	0	[31:24]	AES_KEY_0_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_3_REG[31:24]	AES_KEY_4_REG[31:24]	AES_KEY_5_REG[31:24]	AES_KEY_6_REG[31:24]	AES_KEY_7_REG[31:24]
		[23:16]	AES_KEY_0_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_3_REG[23:16]	AES_KEY_4_REG[23:16]	AES_KEY_5_REG[23:16]	AES_KEY_6_REG[23:16]	AES_KEY_7_REG[23:16]
		[15:8]	AES_KEY_0_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_3_REG[15:8]	AES_KEY_4_REG[15:8]	AES_KEY_5_REG[15:8]	AES_KEY_6_REG[15:8]	AES_KEY_7_REG[15:8]
		[7:0]	AES_KEY_0_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_3_REG[7:0]	AES_KEY_4_REG[7:0]	AES_KEY_5_REG[7:0]	AES_KEY_6_REG[7:0]	AES_KEY_7_REG[7:0]
1	1	[31:24]	AES_KEY_0_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_3_REG[7:0]	AES_KEY_4_REG[7:0]	AES_KEY_5_REG[7:0]	AES_KEY_6_REG[7:0]	AES_KEY_7_REG[7:0]
		[23:16]	AES_KEY_0_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_3_REG[15:8]	AES_KEY_4_REG[15:8]	AES_KEY_5_REG[15:8]	AES_KEY_6_REG[15:8]	AES_KEY_7_REG[15:8]
		[15:8]	AES_KEY_0_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_3_REG[23:16]	AES_KEY_4_REG[23:16]	AES_KEY_5_REG[23:16]	AES_KEY_6_REG[23:16]	AES_KEY_7_REG[23:16]
		[7:0]	AES_KEY_0_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_3_REG[31:24]	AES_KEY_4_REG[31:24]	AES_KEY_5_REG[31:24]	AES_KEY_6_REG[31:24]	AES_KEY_7_REG[31:24]

Note:

- w[0] ~ w[7] are “the first Nk words of the expanded key” as specified in Chapter 5.2 Key Expansion in [NIST FIPS 197](#).
- “Column Bit” specifies the bytes of each word stored in w[0] ~ w[7].



### 19.4.3 Operation Process

#### Single Operation

1. Write 0 to the [AES\\_DMA\\_ENABLE\\_REG](#) register.
2. Initialize registers [AES\\_MODE\\_REG](#), [AES\\_KEY\\_n\\_REG](#), [AES\\_TEXT\\_IN\\_m\\_REG](#), and [AES\\_ENDIAN\\_REG](#).
3. Start operation by writing 1 to the [AES\\_TRIGGER\\_REG](#) register.
4. Wait till the content of the [AES\\_STATE\\_REG](#) register becomes 0, which indicates the operation is completed.
5. Read results from the [AES\\_TEXT\\_OUT\\_m\\_REG](#) register.

#### Consecutive Operations

In consecutive operations, primarily the input [AES\\_TEXT\\_IN\\_m\\_REG](#) and output [AES\\_TEXT\\_OUT\\_m\\_REG](#) registers are being written and read, while the content of [AES\\_DMA\\_ENABLE\\_REG](#), [AES\\_MODE\\_REG](#), [AES\\_KEY\\_n\\_REG](#), and [AES\\_ENDIAN\\_REG](#) is kept unchanged. Therefore, the initialization can be simplified during the consecutive operation.

1. Write 0 to the [AES\\_DMA\\_ENABLE\\_REG](#) register before starting the first operation.
2. Initialize registers [AES\\_MODE\\_REG](#), [AES\\_KEY\\_n\\_REG](#), and [AES\\_ENDIAN\\_REG](#) before starting the first operation.
3. Update the content of [AES\\_TEXT\\_IN\\_m\\_REG](#).
4. Start operation by writing 1 to the [AES\\_TRIGGER\\_REG](#) register.
5. Wait till the content of the [AES\\_STATE\\_REG](#) register becomes 0, which indicates the operation completes.
6. Read results from the [AES\\_TEXT\\_OUT\\_m\\_REG](#) register, and return to Step 3 to continue the next operation.

## 19.5 DMA-AES Working Mode

In the DMA-AES working mode, the AES accelerator supports six block operations including ECB/CBC/OFB/CTR/CFB8/CFB128 as well as GCM operation. Users can choose the operation type for AES accelerator working in the DMA-AES working mode by configuring the [AES\\_BLOCK\\_MODE\\_REG](#) register according to Table 95 below.

**Table 95: Operation Type under DMA-AES Working Mode**

<a href="#">AES_BLOCK_MODE_REG</a> [2:0]	Operation Type
0	ECB (Electronic Codebook)
1	CBC (Cipher Block Chaining)
2	OFB (Output Feedback)
3	CTR (Counter)
4	CFB8 (8-bit Cipher Feedback)
5	CFB128 (128-bit Cipher Feedback)
6	GCM (Galois/Counter Mode)

Users can check the working status of the AES accelerator by inquiring the [AES\\_STATE\\_REG](#) register and comparing the return value against the Table 96 below.

**Table 96: Working Status under DMA-AES Working mode**

<a href="#">AES_STATE_REG</a>	Status	Description
0	IDLE	The AES accelerator is idle.
1	WORK	The AES accelerator is in the middle of an operation.
2	DONE	The AES accelerator completed operations.

When working in the DMA-AES working mode, the AES accelerator supports interrupt on the completion of computation. To enable this function, write 1 to the [AES\\_INT\\_ENA\\_REG](#) register. By default, the interrupt function is enabled. Also, note that the interrupt should be cleared by software after use.

### 19.5.1 Key, Plaintext, and Cipertext

#### Block Operation

During the block operations, the AES Accelerator reads source data (in\_stream) from DMA, and write result data (out\_stream) to DMA after the computation.

- For encryption, DMA reads plaintext from memory, then passes it to AES as source data. After computation, AES passes ciphertext as result data back to DMA to write into memory.
- For decryption, DMA reads ciphertext from memory, then passes it to AES as source data. After computation, AES passes plaintext as result data back to DMA to write into memory.

During block operations, the lengths of the source data and result data are the same. The total computation time is reduced because the DMA data operation and AES computation can happen concurrently.

The length of source data for AES Accelerator under DMA-AES working mode must be 128 bits or the integral multiples of 128 bits. Otherwise, trailing zeros will be added to the original source data, so the length of source data equals to the nearest integral multiples of 128 bits. Please see details in Table 97 below.

Table 97: TEXT-PADDING

Function : TEXT-PADDING()	
<b>Input</b>	: $X$ , bit string.
<b>Output</b>	: $Y = \text{TEXT-PADDING}(X)$ , whose length is the nearest integral multiples of 128 bits.
<b>Steps</b> <p>Let us assume that <math>X</math> is a data-stream that can be split into <math>n</math> parts as following:</p> $X = X_1    X_2    \cdots    X_{n-1}    X_n$ <p>Here, the lengths of <math>X_1, X_2, \cdots, X_{n-1}</math> all equal to 128 bits, and the length of <math>X_n</math> is <math>t</math> (<math>0 &lt; t \leq 127</math>).</p> <p>If <math>t = 0</math>, then</p> $\text{TEXT-PADDING}(X) = X;$ <p>If <math>0 &lt; t \leq 127</math>, define a 128-bit block, <math>X_n^*</math>, and let <math>X_n^* = X_n    0^{128-t}</math>, then</p> $\text{TEXT-PADDING}(X) = X_1    X_2    \cdots    X_{n-1}    X_n^* = X    0^{128-t}$	

### 19.5.2 Endianness

Under the DMA-AES working mode, the transmission of source data and result data for AES Accelerator is solely controlled by DMA. Therefore, the AES Accelerator cannot control the Endianness of the source data and result data, but does have requirement on how these data should be stored in memory and on the length of the data.

For example, let us assume DMA needs to write the following data into memory at address 0x0280.

- Data represented in hexadecimal:
  - 0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20
- Data Length:
  - Equals to 2 blocks.

Then, this data will be stored in memory as shown in Table 98 below.

Table 98: Text Endianness for DMA-AES

Address	Byte	Address	Byte	Address	Byte	Address	Byte
0x0280	0x01	0x0281	0x02	0x0282	0x03	0x0283	0x04
0x0284	0x05	0x0285	0x06	0x0286	0x07	0x0287	0x08
0x0288	0x09	0x0289	0x0A	0x028A	0x0B	0x028B	0x0C
0x028C	0x0D	0x028D	0x0E	0x028E	0x0F	0x028F	0x10
0x0290	0x11	0x0291	0x12	0x0292	0x13	0x0293	0x14
0x0294	0x15	0x0295	0x16	0x0296	0x17	0x0297	0x18
0x0298	0x19	0x0299	0x1A	0x029A	0x1B	0x029B	0x1C
0x029C	0x1D	0x029D	0x1E	0x029E	0x1F	0x029F	0x20

DMA can access both internal memory and PSRAM outside ESP32-S2. When you use DMA to access external PSRAM, please use base addresses that meet the requirements for DMA. When you use DMA to access internal memory, base addresses do not have such requirements. Details can be found in Chapter 7 [DMA Controller](#).

### 19.5.3 Standard Incrementing Function

AES accelerator provides two Standard Incrementing Functions for the CTR block operation, which are INC<sub>32</sub> and INC<sub>128</sub> Standard Incrementing Functions. By setting the [AES\\_INC\\_SEL\\_REG](#) register to 0 or 1, users can choose the INC<sub>32</sub> or INC<sub>128</sub> functions respectively. For details on the Standard Incrementing Function, please see Chapter B.1 The Standard Incrementing Function in [NIST SP 800-38A](#).

### 19.5.4 Block Number

Register [AES\\_BLOCK\\_NUM\\_REG](#) stores the Block Number of plaintext  $P$  or ciphertext  $C$ . The length of this register equals to  $\text{length}(\text{TEXT-PADDING}(P))/128$  or  $\text{length}(\text{TEXT-PADDING}(C))/128$ . The AES Accelerator only uses this register when working in the DMA-AES mode.

### 19.5.5 Initialization Vector

[AES\\_IV\\_MEM](#) is a 16-byte memory, which is only available for AES Accelerator working in block operations. For CBC/OFB/CFB8/CFB128 operations, the [AES\\_IV\\_MEM](#) memory stores the Initialization Vector (IV). For the CTR operation, the [AES\\_IV\\_MEM](#) memory stores the Initial Counter Block (ICB).

Both IV and ICB are 128-bit strings, which can be divided into Byte0, Byte1, Byte2 ... Byte15 (from left to right). [AES\\_IV\\_MEM](#) stores data following the Endianness pattern presented in Table 98, i.e. the most significant (i.e., left-most) byte Byte0 is stored at the lowest address while the least significant (i.e., right-most) byte Byte15 at the highest address.

For more details on IV and ICB, please refer to [NIST SP 800-38A](#).

### 19.5.6 Block Operation Process

1. Write 0 to the [CRYPTO\\_DMA\\_AES\\_SHA\\_SELECT\\_REG](#) register.
2. Configure Crypto DMA chained list and start DMA. For details, please refer to Chapter 7 [DMA Controller](#).
3. Initialize the AES accelerator-related registers:
  - Write 1 to the [AES\\_DMA\\_ENABLE\\_REG](#) register.
  - Configure the [AES\\_INT\\_ENA\\_REG](#) register to enable or disable the interrupt function.
  - Initialize registers [AES\\_MODE\\_REG](#), [AES\\_KEY\\_n\\_REG](#), and [AES\\_ENDIAN\\_REG](#).
  - Select operation type by configuring the [AES\\_BLOCK\\_MODE\\_REG](#) register. For details, see Table 95.
  - Initialize the [AES\\_BLOCK\\_NUM\\_REG](#) register. For details, see Section 19.5.4.
  - Initialize the [AES\\_INC\\_SEL\\_REG](#) register (only needed when AES Accelerator is working under CTR block operation).
  - Initialize the [AES\\_IV\\_MEM](#) memory (This is always needed except for ECB block operation).
4. Start operation by writing 1 to the [AES\\_TRIGGER\\_REG](#) register.
5. Wait for the completion of computation, which happens when the content of [AES\\_STATE\\_REG](#) becomes 2 or the AES interrupt occurs.
6. Check if DMA completes data transmission from AES to memory. At this time, DMA had already written the result data in memory, which can be accessed directly. For details on DMA, please refer to Chapter 7 [DMA Controller](#).

7. Clear interrupt by writing 1 to the [AES\\_INT\\_CLR\\_REG](#) register, if any AES interrupt occurred during the computation.
8. Release the AES Accelerator by writing 0 to the [AES\\_DMA\\_EXIT\\_REG](#) register. After this, the content of the [AES\\_STATE\\_REG](#) register becomes 0. Note that, you can release DMA earlier, but only after Step 5 is completed.

### 19.5.7 GCM Operation Process

1. Write 0 to the [CRYPTO\\_DMA\\_AES\\_SHA\\_SELECT\\_REG](#) register.
2. Configure Crypto DMA chained list and start DMA. For details on DMA, please refer to Chapter 7 [DMA Controller](#).
3. Initialize the AES accelerator-related registers:
  - Write 1 to the [AES\\_DMA\\_ENABLE\\_REG](#) register.
  - Configure the [AES\\_INT\\_ENA\\_REG](#) register to enable or disable the interrupt function.
  - Initialize registers [AES\\_MODE\\_REG](#) and [AES\\_KEY\\_n\\_REG](#) [AES\\_ENDIAN\\_REG](#).
  - Write 6 to the [AES\\_BLOCK\\_MODE\\_REG](#) register.
  - Initialize the [AES\\_BLOCK\\_NUM\\_REG](#) register. Details about this register are described in Section 19.5.4.
  - Initialize the [AES\\_AAD\\_BLOCK\\_NUM\\_REG](#) register. Details about this register are described in Section 19.6.4.
  - Initialize the [AES\\_REMAINDER\\_BIT\\_NUM\\_REG](#) register. Details about this register is described in Section 19.6.5.
4. Start operation by writing 1 to the [AES\\_TRIGGER\\_REG](#) register.
5. Wait for the completion of computation, which happens when the content of [AES\\_STATE\\_REG](#) becomes 2. For details on the working status of AES Accelerator, please refer to Table 96. At this step, no interrupt occurs.
6. Obtain the  $H$  value from the [AES\\_H\\_MEM](#) memory.
7. Generate  $J_0$  and write it to the [AES\\_J0\\_MEM](#) memory.
8. Continue operating by writing 1 to the [AES\\_CONTINUE\\_REG](#) register.
9. Wait for the completion of computation, which happens when the content of [AES\\_STATE\\_REG](#) becomes 2 or the AES interrupt occurs. For details on the working status of AES Accelerator, please refer to Table 96.
10. Obtain  $T_0$  by reading [AES\\_T0\\_MEM](#), which is already ready at this step.
11. Check if DMA completes data transmission from AES to memory. At this time, DMA had already written the result data in memory, which can be accessed directly. For details on DMA, please refer to Chapter 7 [DMA Controller](#).
12. Clear interrupt by writing 1 to the [AES\\_INT\\_CLR\\_REG](#) register, if any AES interrupt occurred during the computation.
13. Exit DMA by writing 1 to the [AES\\_DMA\\_EXIT\\_REG](#) register. After this, the content of the [AES\\_STATE\\_REG](#) becomes 0. Note that, you can exit DMA earlier, but only after Step 9 is completed.

## 19.6 GCM Algorithm

ESP32-S2's AES accelerator fully supports GCM Algorithm. In reality, the  $AAD$ ,  $C$  and  $P$  that are longer than  $2^{32}-1$  bits are seldom used. Therefore, we specify that the length of  $AAD$ ,  $C$  and  $P$  should be no longer than  $2^{32}-1$  here. Figure 19-12 below demonstrates how GCM encryption is implemented in the AES Accelerator of ESP32-S2.

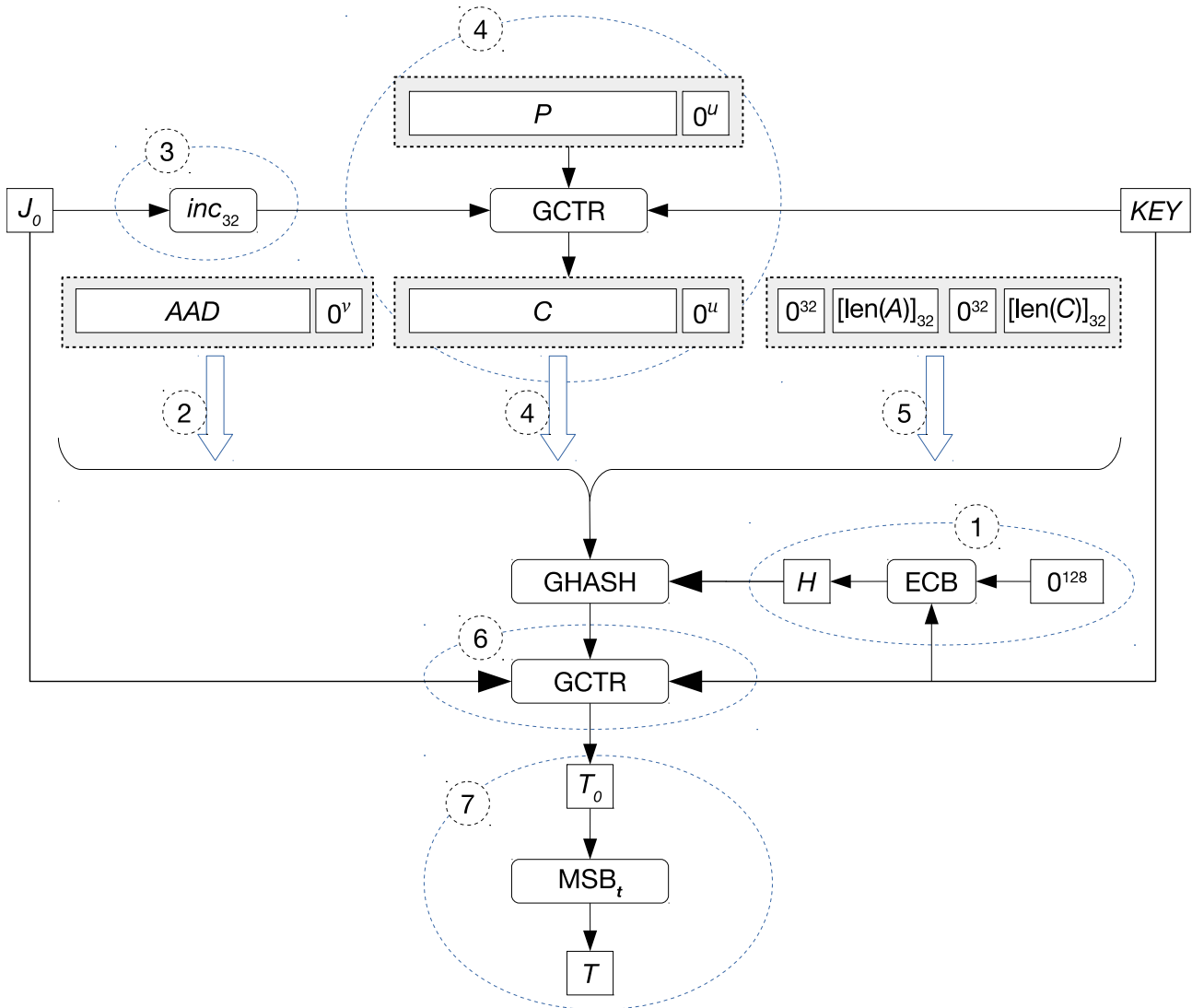


Figure 19-12. GCM Encryption Process

GCM encryption is implemented as follows:

1. Hardware executes the ECB Algorithm to obtain the Hash subkey  $H$ , which is needed in the Hash computation.
2. Hardware executes the GHASH Algorithm to perform Hash computation with the padded  $AAD$ .
3. Hardware gets ready for CTR encryption by obtaining the result of applying Standard Incrementing Function  $INC_{32}$  to  $J_0$ .
4. Hardware executes the GCTR Algorithm to encrypt the padded plaintext  $P$ , then executes the GHASH Algorithm to perform Hash computation on the padded ciphertext  $C$ .
5. Hardware executes the GHASH Algorithm to perform Hash computation on AAD Blocks, obtaining a

128-bit Hash result.

6. Hardware executes the GCTR Algorithm to encrypt  $J_0$ , obtaining  $T_0$ .
7. Software obtains the result  $T_0$  from hardware, and execute  $MSB_t$  Algorithm to obtain the final result Authenticated Tag  $T$ .

The only difference between GCM decryption and GCM encryption lies in Step 4 in Figure 19-12. To be more specific, instead of executing GCTR Algorithm to encrypt the padded plaintext, the AES Accelerator executes the same Algorithm to decrypt the padded ciphertext in GCM decryption. For details, please see [NIST SP 800-38D](#).

### 19.6.1 Hash Subkey

During GCM operation, the Hash subkey  $H$  is a 128-bit value computed by hardware, which is demonstrated in Step 1 in Figure 19-12. Also you can find more information about Hash subkey at “Step 1. Let  $H = CIPH_K(0^{128})$ ” in Chapter 7 GCM Specification of [NIST SP 800-38D](#).

Just like all other Endianness, the Hash subkey  $H$  is stored in the [AES\\_H\\_MEM](#) memory with its most significant (i.e., left-most) byte Byte0 stored at the lowest address and least significant (i.e., right-most) byte Byte15 at the highest address. For details, see Table 98.

### 19.6.2 $J_0$

The  $J_0$  is a 128-bit value computed by hardware, which is required during Step 3 and Step 6 of the GCM process in Figure 19-12. For details on the generation of  $J_0$ , please see Chapter 7 GCM Specification in [NIST SP 800-38D](#) Specification.

The  $J_0$  is stored in the [AES\\_JO\\_MEM](#) memory as Endianness. Just like all other Endianness, its most significant (i.e., left-most) byte Byte0 is stored at the lowest address in the memory while least significant (i.e., right-most) byte Byte15 at the highest address. For details, see Table 98.

### 19.6.3 Authenticated Tag

Authenticated Tag (Tag for short) is one of the key results of GCM computation, which is demonstrated in Step 7 of Figure 19-12. The value of Tag is determined by the length of Authenticated Tag  $t$  ( $1 \leq t \leq 128$ ):

- When  $t = 128$ , the value of Tag equals to  $T_0$ , a 128-bit string that is stored in the [AES\\_T0\\_MEM](#) as Endianness. Just like all other Endianness, its most significant (i.e., left-most) byte Byte0 is stored at the lowest address in the memory while least significant (i.e., right-most) byte Byte15 at the highest address. For details, see Table 98.
- When  $1 \leq t < 128$ , the value of Tag equals to the  $t$  most significant (i.e., left-most) bits of  $T_0$ . In this case, Tag is represented as  $MSB_t(T_0)$ , which returns the  $t$  most significant bits of  $T_0$ . For example,  $MSB_4(111011010) = 1110$  and  $MSB_5(11010011010) = 11010$ . For details on the  $MSB_t()$  function, please refer to Chapter 6 Mathematical Components of GCM in the [NIST SP 800-38D](#) specification.

### 19.6.4 AAD Block Number

Register [AES\\_AAD\\_BLOCK\\_NUM\\_REG](#) stores the Block Number of Additional Authenticated Data (AAD). The length of this register equals to  $\text{length}(\text{TEXT-PADDING}(\text{AAD}))/128$ . AES Accelerator only uses this register when working in the DMA-AES mode.

### 19.6.5 Remainder Bit Number

Register [AES\\_REMAINDER\\_BIT\\_NUM\\_REG](#) stores the Remainder Bit Number, which indicates the number of effective bits of incomplete blocks in plaintext/ciphertext. The value stored in this register equals to  $\text{length}(P)\%128$  or  $\text{length}(C)\%128$ . AES Accelerator only uses this register when working in the DMA-AES mode.

Register [AES\\_REMAINDER\\_BIT\\_NUM\\_REG](#) does not affect the results of plaintext or ciphertext, but does impact the value of  $T_0$ , therefore the Tag value too.

The GCM Algorithm can be viewed as the combination of GCTR operation and GHASH operation, among which, the GCTR performs the encryption and decryption, while the GHASH solves the Tag.

Note that the [AES\\_REMAINDER\\_BIT\\_NUM\\_REG](#) register is only effective for GCM encryption. To be more specific:

- For GCM encryption, the Hardware firstly computes  $C$ , then passes it in the form of **TEXT-PADDING( $C$ )** as the input of GHASH operation. In this case, hardware determines how many trailing “0” should be added based on the content of [AES\\_REMAINDER\\_BIT\\_NUM\\_REG](#).
- For GCM decryption, the padding is completed with the **TEXT-PADDING( $C$ )** function. In this case, the [AES\\_REMAINDER\\_BIT\\_NUM\\_REG](#) register is not effective.

## 19.7 Base Address

Users can access AES with two base addresses, which can be seen in Table 99. For more information about accessing peripherals from different buses please see Chapter 1 *System and Memory*.

**Table 99: AES Accelerator Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F43A000
PeriBUS2	0x6003A000

## 19.8 Memory Summary

Both the starting address and ending address in the following table are relative to AES base addresses provided in Section 19.7.

**Table 100: AES Accelerator Memory Blocks**

Name	Description	Size (byte)	Starting Address	Ending Address	Access
<a href="#">AES_IV_MEM</a>	Memory IV	16 bytes	0x0050	0x005F	R/W
<a href="#">AES_H_MEM</a>	Memory H	16 bytes	0x0060	0x006F	RO
<a href="#">AES_J0_MEM</a>	Memory J0	16 bytes	0x0070	0x007F	R/W
<a href="#">AES_T0_MEM</a>	Memory T0	16 bytes	0x0080	0x008F	RO

## 19.9 Register Summary

The addresses in the following table are relative to AES base addresses provided in Section 19.7.

Name	Description	Address	Access
<b>Key Registers</b>			
<a href="#">AES_KEY_0_REG</a>	AES key register 0	0x0000	R/W



Name	Description	Address	Access
<a href="#">AES_KEY_1_REG</a>	AES key register 1	0x0004	R/W
<a href="#">AES_KEY_2_REG</a>	AES key register 2	0x0008	R/W
<a href="#">AES_KEY_3_REG</a>	AES key register 3	0x000C	R/W
<a href="#">AES_KEY_4_REG</a>	AES key register 4	0x0010	R/W
<a href="#">AES_KEY_5_REG</a>	AES key register 5	0x0014	R/W
<a href="#">AES_KEY_6_REG</a>	AES key register 6	0x0018	R/W
<a href="#">AES_KEY_7_REG</a>	AES key register 7	0x001C	R/W
<b>TEXT_IN Registers</b>			
<a href="#">AES_TEXT_IN_0_REG</a>	Source data register 0	0x0020	R/W
<a href="#">AES_TEXT_IN_1_REG</a>	Source data register 1	0x0024	R/W
<a href="#">AES_TEXT_IN_2_REG</a>	Source data register 2	0x0028	R/W
<a href="#">AES_TEXT_IN_3_REG</a>	Source data register 3	0x002C	R/W
<b>TEXT_OUT Registers</b>			
<a href="#">AES_TEXT_OUT_0_REG</a>	Result data register 0	0x0030	RO
<a href="#">AES_TEXT_OUT_1_REG</a>	Result data register 1	0x0034	RO
<a href="#">AES_TEXT_OUT_2_REG</a>	Result data register 2	0x0038	RO
<a href="#">AES_TEXT_OUT_3_REG</a>	Result data register 3	0x003C	RO
<b>Configuration Registers</b>			
<a href="#">AES_MODE_REG</a>	AES working mode configuration register	0x0040	R/W
<a href="#">AES_ENDIAN_REG</a>	Endian configuration register	0x0044	R/W
<a href="#">AES_DMA_ENABLE_REG</a>	DMA enable register	0x0090	R/W
<a href="#">AES_BLOCK_MODE_REG</a>	Block operation type register	0x0094	R/W
<a href="#">AES_BLOCK_NUM_REG</a>	Block number configuration register	0x0098	R/W
<a href="#">AES_INC_SEL_REG</a>	Standard incrementing function register	0x009C	R/W
<a href="#">AES_AAD_BLOCK_NUM_REG</a>	AAD block number configuration register	0x00A0	R/W
<a href="#">AES_REMAINDER_BIT_NUM_REG</a>	Remainder bit number of plaintext/ciphertext	0x00A4	R/W
<b>Controlling / Status Registers</b>			
<a href="#">AES_TRIGGER_REG</a>	Operation start controlling register	0x0048	WO
<a href="#">AES_STATE_REG</a>	Operation status register	0x004C	RO
<a href="#">AES_CONTINUE_REG</a>	Operation continue controlling register	0x00A8	WO
<a href="#">AES_DMA_EXIT_REG</a>	Operation exit controlling register	0x00B8	WO
<b>Interrupt Registers</b>			
<a href="#">AES_INT_CLR_REG</a>	DMA-AES interrupt clear register	0x00AC	WO
<a href="#">AES_INT_ENA_REG</a>	DMA-AES interrupt enable register	0x00B0	R/W

## 19.10 Registers

Register 19.1: AES\_KEY\_ *n* \_REG (*n*: 0-7) (0x0000+4\**n*)

31	0
0x00000000	
Reset	

AES\_KEY\_ *n* \_REG (*n*: 0-7) Stores AES keys. (R/W)

Register 19.2: AES\_TEXT\_IN\_ *m* \_REG (*m*: 0-3) (0x0020+4\**m*)

31	0
0x00000000	
Reset	

AES\_TEXT\_IN\_ *m* \_REG (*m*: 0-3) Stores the source data when the AES Accelerator operates in the Typical AES working mode. (R/W)

Register 19.3: AES\_TEXT\_OUT\_ *m* \_REG (*m*: 0-3) (0x0030+4\**m*)

31	0
0x00000000	
Reset	

AES\_TEXT\_OUT\_ *m* \_REG (*m*: 0-3) Stores the result data when the AES Accelerator operates in the Typical AES working mode. (RO)

Register 19.4: AES\_MODE\_REG (0x0040)

(reserved)		AES_MODE	
31	3	2	0
0x00000000		0	
		Reset	

AES\_MODE Defines the operation type of the AES Accelerator operating under the Typical AES working mode. For details, see Table 89. (R/W)

### Register 19.5: AES\_ENDIAN\_REG (0x0044)

(reserved)										AES_ENDIAN								
31										6	5	0						
0x00000000											0	0	0	0	0	0	0	Reset

<b>AES_ENDIAN</b>	Defines the endianness of input and output texts. For details, please see Table 91.
(R/W)	

### Register 19.6: AES\_DMA\_ENABLE\_REG (0x0090)

31	1	0	
0x00000000		0	Reset

**AES\_DMA\_ENABLE** Defines the working mode of the AES Accelerator. For details, see Table 88.

(R/W)

### Register 19.7: AES\_BLOCK\_MODE\_REG (0x0094)

(reserved)		AES_BLOCK_MODE	
31	3	2	0
0x00000000			0
			Reset

**AES\_BLOCK\_MODE** Defines the operation type of the AES Accelerator operating under the DMA-AES working mode. For details, see Table 95. (R/W)

### Register 19.8: AES\_BLOCK\_NUM\_REG (0x0098)

31	0
0x00000000	
Reset	

**AES\_BLOCK\_NUM** Stores the Block Number of plaintext or ciphertext when the AES Accelerator operates under the DMA-AES working mode. For details, see Section 19.5.4. (R/W)

**Register 19.9: AES\_INC\_SEL\_REG (0x009C)**

(reserved)																															AES_INC_SEL	
31																														1	0	Reset
0x00000000																														0		

**AES\_INC\_SEL** Defines the Standard Incrementing Function for CTR block operation. Set this bit to 0 or 1 to choose INC<sub>32</sub> or INC<sub>128</sub>. (R/W)

**Register 19.10: AES\_AAD\_BLOCK\_NUM\_REG (0x00A0)**

31																0	Reset
0x00000000																	

**AES\_AAD\_BLOCK\_NUM** Stores the ADD Block Number for the GCM operation. (R/W) For details, see Section 19.6.4.

**Register 19.11: AES\_REMAINDER\_BIT\_NUM\_REG (0x00A4)**

(reserved)															AES_REMAINDER_BIT_NUM		
31														7	6	0	Reset
0x00000000														0			

**AES\_REMAINDER\_BIT\_NUM** Stores the Remainder Bit Number for the GCM operation. For details, see Section 19.6.5. (R/W)

**Register 19.12: AES\_TRIGGER\_REG (0x0048)**

(reserved)																															AES_TRIGGER	
31																															1	0
0x00000000																															x	Reset

**AES\_TRIGGER** Set this bit to 1 to start AES operation. (WO)

**Register 19.13: AES\_STATE\_REG (0x004C)**

(reserved)		AES_STATE	
31	2	1	0
0x00000000		0x0	Reset

**AES\_STATE** Stores the working status of the AES Accelerator. For details, see Table 90 for Typical AES working mode and Table 96 for DMA AES working mode. (RO)

**Register 19.14: AES\_CONTINUE\_REG (0x00A8)**

(reserved)		AES_CONTINUE	
31	1	0	
0x00000000		x	Reset

**AES\_CONTINUE** Set this bit to 1 to continue AES operation. (WO)

**Register 19.15: AES\_DMA\_EXIT\_REG (0x00B8)**

(reserved)		AES_DMA_EXIT	
31	1	0	
0x00000000		x	Reset

**AES\_DMA\_EXIT** Set this bit to 1 to exit AES operation. This register is only effective for DMA-AES operation. (WO)

**Register 19.16: AES\_INT\_CLR\_REG (0x00AC)**

(reserved)		AES_INT_CLR	
31	1	0	
0x00000000		x	Reset

**AES\_INT\_CLR** Set this bit to 1 to clear AES interrupt. (WO)

Register 19.17: AES\_INT\_ENA\_REG (0x00B0)

(reserved)		AES_INT_ENA	
31	1	0	
0x00000000		0	Reset

**AES\_INT\_ENA** Set this bit to 1 to enable AES interrupt and 0 to disable interrupt. (R/W)

## 20. SHA Accelerator

### 20.1 Introduction

ESP32-S2 integrates an SHA accelerator, which is a hardware device that speeds up SHA algorithm significantly, compared to SHA algorithm implemented solely in software. The SHA accelerator integrated in ESP32-S2 has two working modes, which are [Typical SHA](#) and [DMA-SHA](#).

### 20.2 Features

The following functionality is supported:

- All the hash algorithms introduced in [FIPS PUB 180-4 Spec](#).
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
  - SHA-512/224
  - SHA-512/256
  - SHA-512/*t*
- Two working modes
  - Typical SHA
  - DMA-SHA
- Interleave function when working in Typical SHA working mode
- Interrupt function when working in DMA-SHA working mode

### 20.3 Working Modes

The SHA accelerator integrated in ESP32-S2 has two working modes: [Typical SHA](#) and [DMA-SHA](#).

- [Typical SHA Working Mode](#): all the data is written and read via CPU directly.
- [DMA-SHA Working Mode](#): all the data is read via crypto DMA. That is, users can configure the DMA controller to read all the data needed for hash operation, thus releasing CPU for completing other tasks.

Users can start the SHA accelerator with different working modes by configuring registers [SHA\\_START\\_REG](#) and [SHA\\_DMA\\_START\\_REG](#). For details, please see Table [102](#).

**Table 102: SHA Accelerator Working Mode**

Working Mode	Configuration Method
Typical SHA	Set <a href="#">SHA_START_REG</a> to 1
DMA-SHA	Set <a href="#">SHA_DMA_START_REG</a> to 1

Users can choose hash algorithms by configuring the [SHA\\_MODE\\_REG](#) register. For details, please see Table 103.

**Table 103: SHA Hash Algorithm**

Hash Algorithm	<a href="#">SHA_MODE_REG</a> Configuration
SHA-1	0
SHA-224	1
SHA-256	2
SHA-384	3
SHA-512	4
SHA-512/224	5
SHA-512/256	6
SHA-512/ <i>t</i>	7

**Notice:**

ESP32-S2's [Digital Signature](#) and HMAC modules also call the SHA accelerator. Therefore, users cannot access the SHA accelerator when these modules are working.

## 20.4 Function Description

SHA accelerator can generate the message digest via two steps: [Preprocessing](#) and [Hash operation](#).

### 20.4.1 Preprocessing

Preprocessing consists of three steps: [padding the message](#), [parsing the message into message blocks](#) and [setting the initial hash value](#).

#### 20.4.1.1 Padding the Message

The SHA accelerator can only process message blocks of 512 or 1024 bits, depending on the algorithm. Thus, all the messages should be padded to a multiple of 512 or 1024 bits before the hash computation.

Suppose that the length of the message  $M$  is  $m$  bits. Then  $M$  shall be padded as introduced below:

- **SHA-1, SHA-224 and SHA-256**

1. First, append the bit "1" to the end of the message;
2. Second, append  $k$  zero bits, where  $k$  is the smallest, non-negative solution to the equation  $m + 1 + k \equiv 448 \pmod{512}$ ;
3. Last, append the 64-bit block that is equal to the number  $m$  expressed using a binary representation.



- **SHA-384, SHA-512, SHA-512/224, SHA-512/256 and SHA-512/t**

1. First, append the bit “1” to the end of the message;
2. Second, append  $k$  zero bits, where  $k$  is the smallest, non-negative solution to the equation  $m + 1 + k \equiv 896 \bmod 1024$ ;
3. Last, append the 128-bit block that is equal to the number  $m$  expressed using a binary representation.

For more details, please refer to Section “5.1 Padding the Message” in [FIPS PUB 180-4 Spec.](#)

### 20.4.1.2 Parsing the Message

The message and its padding must be parsed into  $N$  512-bit or 1024-bit blocks.

- For **SHA-1, SHA-224 and SHA-256**: the message and its padding are parsed into  $N$  512-bit blocks,  $M^{(1)}$ ,  $M^{(2)}$ , ...,  $M^{(N)}$ . Since the 512 bits of the input block may be expressed as sixteen 32-bit words, the first 32 bits of message block  $i$  are denoted  $M_0^{(i)}$ , the next 32 bits are  $M_1^{(i)}$ , and so on up to  $M_{15}^{(i)}$ .
- For **SHA-384, SHA-512, SHA-512/224, SHA-512/256 and SHA-512/t**: the message and its padding are parsed into  $N$  1024-bit blocks. Since the 1024 bits of the input block may be expressed as sixteen 64-bit words, the first 64 bits of message block  $i$  are denoted  $M_0^{(i)}$ , the next 64 bits are  $M_1^{(i)}$ , and so on up to  $M_{15}^{(i)}$ .

In **Typical SHA** working mode, all the message blocks are written into the **SHA\_M\_n\_REG**, following the rules below:

- For **SHA-1, SHA-224 and SHA-256**:  $M_0^{(i)}$  is stored in **SHA\_M\_0\_REG**,  $M_1^{(i)}$  stored in **SHA\_M\_1\_REG**, ..., and  $M_{15}^{(i)}$  stored in **SHA\_M\_15\_REG**.
- For **SHA-384, SHA-512, SHA-512/224 and SHA-512/256**: the most significant 32 bits and the least significant 32 bits of  $M_0^{(i)}$  are stored in **SHA\_M\_0\_REG** and **SHA\_M\_1\_REG**, respectively, ..., the most significant 32 bits and the least significant 32 bits of  $M_{15}^{(i)}$  are stored in **SHA\_M\_30\_REG** and **SHA\_M\_31\_REG**, respectively.

**Note:**

For more information about “message block”, please refer to Section “2.1 Glossary of Terms and Acronyms” in [FIPS PUB 180-4 Spec.](#)

In **DMA-SHA** working mode, please complete the following configuration:

1. Create an external linked list;
2. Configure this linked list based on the instruction described in Chapter *DMA Controller*, including but not limited to assigning the starting address of the input message to the buffer address pointer of the linked list;
3. Configure the **CRYPTO\_DMA\_OUTLINK\_ADDR** to the first out-link linked list;
4. Write 1 to register **CRYPTO\_DMA\_OUTLINK\_START**, so the DMA starts to move data;
5. Write 1 to register **CRYPTO\_DMA\_AES\_SHA\_SELECT\_REG**, so the SHA accelerator gets to use the DMA resource shared by AES and SHA accelerators.

### 20.4.1.3 Initial Hash Value

Before hash computation begins for each of the secure hash algorithms, the initial Hash value  $H^{(0)}$  must be set based on different algorithms, among which the SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256 algorithms use the initial Hash values (constant C) stored in the hardware.

However, SHA-512/ $t$  requires a distinct initial hash value for each operation for a given value of  $t$ . Simply put, SHA-512/ $t$  is the generic name for a  $t$ -bit hash function based on SHA-512 whose output is truncated to  $t$  bits.  $t$  is any positive integer without a leading zero such that  $t < 512$ , and  $t$  is not 384. The initial hash value for SHA-512/ $t$  for a given value of  $t$  can be calculated by performing SHA-512 from hexadecimal representation of the string "SHA-512/ $t$ ". It's not hard to observe that when determining the initial hash values for SHA-512/ $t$  algorithms with different  $t$ , the only difference lies in the value of  $t$ .

Therefore, we have specially developed the following simplified method to calculate the initial hash value for SHA-512/ $t$ :

1. **Generate  $t\_string$  and  $t\_length$ :**  $t\_string$  is a 32-bit data that stores the input message of  $t$ .  $t\_length$  is a 7-bit data that stores the length of the input message. The  $t\_string$  and  $t\_length$  are generated in methods described below, depending on the value of  $t$ :

- If  $1 \leq t \leq 9$ , then  $t\_length = 7'h48$  and  $t\_string$  is padded in the following format:

$8'h3t_0$	$1'b1$	$23'b0$
-----------	--------	---------

where  $t_0 = t$ .

For example, if  $t = 8$ , then  $t_0 = 8$  and  $t\_string = 32'h38800000$ .

- If  $10 \leq t \leq 99$ , then  $t\_length = 7'h50$  and  $t\_string$  is padded in the following format:

$8'h3t_1$	$8'h3t_0$	$1'b1$	$15'b0$
-----------	-----------	--------	---------

where,  $t_0 = t \% 10$  and  $t_1 = t / 10$ .

For example, if  $t = 56$ , then  $t_0 = 6$ ,  $t_1 = 5$ , and  $t\_string = 32'h35368000$ .

- If  $100 \leq t < 512$ , then  $t\_length = 7'h58$  and  $t\_string$  is padded in the following format:

$8'h3t_2$	$8'h3t_1$	$8'h3t_0$	$1'b1$	$7'b0$
-----------	-----------	-----------	--------	--------

where,  $t_0 = t \% 10$ ,  $t_1 = (t / 10) \% 10$ , and  $t_2 = t / 100$ .

For example, if  $t = 231$ , then  $t_0 = 1$ ,  $t_1 = 3$ ,  $t_2 = 2$ , and  $t\_string = 32'h32333180$ .

2. **Initialize relevant registers:** Initialize [SHA\\_T\\_STRING\\_REG](#) and [SHA\\_T\\_LENGTH\\_REG](#) with the generated  $t\_string$  and  $t\_length$  in the previous step.
3. **Obtain initial hash value:** Set the [SHA\\_MODE\\_REG](#) register to 7. Set the [SHA\\_START\\_REG](#) register to 1 to start the SHA accelerator. Then poll register [SHA\\_BUSY\\_REG](#) until the content of this register becomes 0, indicating the calculation of initial hash value is completed.

Please note that the initial value for SHA-512/ $t$  can be also calculated according to the Section "5.3.6 SHA-512/ $t$ " in [FIPS PUB 180-4 Spec](#), that is performing SHA-512 operation (with its initial hash value set to the result of 8-bitwise XOR operation of C and 0xa5) from the hexadecimal representation of the string "SHA-512/ $t$ ".

## 20.4.2 Hash Computation Process

After the preprocessing, the ESP32-S2 SHA accelerator starts to hash a message  $M$  and generates message digest of different lengths, depending on different hash algorithms. As described above, the ESP32-S2 SHA accelerator supports two working modes, which are [Typical SHA](#) and [DMA-SHA](#). The operation process for the SHA accelerator under two working modes is described in the following subsections.

### 20.4.2.1 Typical SHA Process

ESP32-S2 SHA accelerator supports “interleave” functionality when working under Typical SHA mode:

- **Type “alone”:** Users do not insert any new computation before the SHA accelerator completes all the message blocks.
- **Type “interleave”:** Users can insert new computations (both Typical SHA task and DMA-SHA task) every time the SHA accelerator completes one message block. To be more specific, users can store the message digest in registers [SHA\\_H\\_n\\_REG](#) after completing each message block, and assign the accelerator with other higher priority tasks. After the inserted task completes, users can put the message digest stored back to registers [SHA\\_H\\_n\\_REG](#), and resume the accelerator with the previously paused computation.

#### Typical SHA Process (except for SHA-512/t)

1. Select a hash algorithm.
  - Select a hash algorithm by configuring the [SHA\\_MODE\\_REG](#) register. For details, please refer to Table [103](#).
2. Process the current message block.
  - (a) Write the current message block in registers [SHA\\_M\\_n\\_REG](#);
  - (b) Start the SHA accelerator <sup>1</sup>:
    - If this is the first time to execute this step, set the [SHA\\_START\\_REG](#) register to 1 to start the SHA accelerator. In this case, the accelerator uses the initial hash value stored in hardware for a given algorithm configured in Step 1 to start the computation;
    - If this is not the first time to execute this step, set the [SHA\\_CONTINUE\\_REG](#) register to 1 to start the SHA accelerator. In this case, the accelerator uses the hash value stored in the [SHA\\_H\\_n\\_REG](#) register to start computation.
  - (c) Poll register [SHA\\_BUSY\\_REG](#) until the content of this register becomes 0, indicating the accelerator has completed the computation for the current message block and now is in the “idle” status. Then, go to step 3.
3. Decide if you want to insert other computations.
  - If yes, please get ready for handing over the SHA accelerator to the new task:
    - (a) Read and store the hash algorithm selected for the current computation stored in the [SHA\\_MODE\\_REG](#) register;
    - (b) Read and store the message digest stored in registers [SHA\\_H\\_n\\_REG](#);
    - (c) Last, please go to perform the inserted computation. For the detailed process of the inserted computation, please refer to [Typical SHA](#) or [DMA-SHA](#), depending on the working mode.

- Otherwise, please continue to execute Step 4.
4. Decide if you have more message blocks following the previous computation:
    - If yes, please go back to 2.
    - Otherwise, go to Step 5.
  5. Decide if you need to return the SHA accelerator to a previous computation (i.e., decide whether the current task is an inserted task or not):
    - If yes, please get ready to return the SHA accelerator for the previous computation:
      - (a) Write the previously stored hash algorithm back to register `SHA_MODE_REG`;
      - (b) Write the previously stored message digest back to registers `SHA_H_n_REG`;
      - (c) Then, go to Step 2.
    - Otherwise, there is no need to return SHA Control. Therefore, please go to Step 6 directly.
  6. Obtain the message digest:
    - Read the message digest from registers `SHA_H_n_REG`.

#### Typical SHA Process (SHA-512/t)

1. Select a hash algorithm.
  - Select SHA-512/t algorithm by configuring the `SHA_MODE_REG` register to 7.
2. Calculate the initial hash value.
  - (a) Calculate `t_string` and `t_length` and initialize `SHA_T_STRING_REG` and `SHA_T_LENGTH_REG` with the generated `t_string` and `t_length`. For details, please refer to Section 20.4.1.3.
  - (b) Set the `SHA_START_REG` register to 1 to start the SHA accelerator.
  - (c) Poll register `SHA_BUSY_REG` until the content of this register becomes 0, indicating the calculation of initial hash value is completed.
3. Process the current message block.
  - (a) Write the current message block in registers `SHA_M_n_REG`;
  - (b) Start the SHA accelerator <sup>1</sup>:
    - Set the `SHA_CONTINUE_REG` register to 1 to start the SHA accelerator. In this case, the accelerator uses the hash value stored in the `SHA_H_n_REG` register to start computation.
  - (c) Poll register `SHA_BUSY_REG` until the content of this register becomes 0, indicating the accelerator has completed the computation for the current message block and now is in the “idle” status. Then, go to step 4.
4. Decide if you want to insert other computations.
  - If yes, please get ready for handing over the SHA accelerator to the new task:
    - (a) Read and store the hash algorithm selected for the current computation stored in the `SHA_MODE_REG` register;
    - (b) Read and store the message digest stored in registers `SHA_H_n_REG`;

- (c) Last, please go to perform the inserted computation. For the detailed process of the inserted computation, please refer to [Typical SHA](#) or [DMA-SHA](#), depending on the working mode.
  - Otherwise, please continue to execute Step 5.
5. Decide if you have more message blocks following the previous computation:
    - If yes, please go back to 3.
    - Otherwise, go to Step 6.
  6. Decide if you need to return the SHA accelerator to a previous computation (i.e., decide whether the current task is an inserted task or not):
    - If yes, please get ready to return the SHA accelerator for the previous computation:
      - (a) Write the previously stored hash algorithm back to register [SHA\\_MODE\\_REG](#);
      - (b) Write the previously stored message digest back to registers [SHA\\_H\\_n\\_REG](#);
      - (c) Then, go to Step 3.
    - Otherwise, there is no need to return SHA Control. Therefore, please go to Step 7 directly.
  7. Obtain the message digest:
    - Read the message digest from registers [SHA\\_H\\_n\\_REG](#).

**Note:**

1. In Step 2b, the software can also write the next message block (to be processed) in registers [SHA\\_M\\_n\\_REG](#), if any, while the hardware starts SHA computation, to save time.

### 20.4.2.2 DMA-SHA Process

ESP32-S2 SHA accelerator does not support type “interleave” computation, which means you cannot insert new computation before the whole DMA-SHA process completes. In this mode, users who need task insertion are recommended to divide your message blocks and perform several DMA-SHA computations, instead of trying to compute all the messages in one go.

In contrast to the Typical SHA working mode, when the SHA accelerator is working under the DMA-SHA mode, all data read are completed via crypto DMA.

Therefore, users are required to configure the DMA controller as instructed in Subsection [20.4.1.2](#).

#### DMA-SHA process (except SHA-512/t)

1. Select a hash algorithm.
  - Select a hash algorithm by configuring the [SHA\\_MODE\\_REG](#) register. For details, please refer to Table [103](#).
2. Configure the [SHA\\_INT\\_ENA\\_REG](#) register to enable or disable interrupt (Set 1 to enable).
3. Configure the number of message blocks.
  - Write the number of message blocks  $M$  to the [SHA\\_DMA\\_BLOCK\\_NUM\\_REG](#) register.
4. Start the DMA-SHA computation.

- If the current DMA-SHA computation follows a previous computation, firstly write the message digest from the previous computation to registers [SHA\\_H\\_n\\_REG](#), then write 1 to register [SHA\\_DMA\\_CONTINUE\\_REG](#) to start SHA accelerator;
- Otherwise, write 1 to register [SHA\\_DMA\\_START\\_REG](#) to start the accelerator.

5. Wait till the completion of the DMA-SHA computation, which happens when:

- The content of [SHA\\_BUSY\\_REG](#) register becomes 0, or
- An SHA interrupt occurs. In this case, please clear interrupt by writing 1 to the [SHA\\_INT\\_CLEAR\\_REG](#) register.

6. Obtain the message digest:

- Read the message digest from registers [SHA\\_H\\_n\\_REG](#).

#### **DMA-SHA process for SHA-512/t**

1. Select a hash algorithm.

- Select SHA-512/t algorithm by configuring the [SHA\\_MODE\\_REG](#) register to 7.

2. Configure the [SHA\\_INT\\_ENA\\_REG](#) register to enable or disable interrupt (Set 1 to enable).

3. Calculate the initial hash value.

- (a) Calculate `t_string` and `t_length` and initialize [SHA\\_T\\_STRING\\_REG](#) and [SHA\\_T\\_LENGTH\\_REG](#) with the generated `t_string` and `t_length`. For details, please refer to Section 20.4.1.3.
- (b) Set the [SHA\\_START\\_REG](#) register to 1 to start the SHA accelerator.
- (c) Poll register [SHA\\_BUSY\\_REG](#) until the content of this register becomes 0, indicating the calculation of initial hash value is completed.

4. Configure the number of message blocks.

- Write the number of message blocks  $M$  to the [SHA\\_DMA\\_BLOCK\\_NUM\\_REG](#) register.

5. Start the DMA-SHA computation.

- Write 1 to register [SHA\\_DMA\\_CONTINUE\\_REG](#) to start the accelerator.

6. Wait till the completion of the DMA-SHA computation, which happens when:

- The content of [SHA\\_BUSY\\_REG](#) register becomes 0, or
- An SHA interrupt occurs. In this case, please clear interrupt by writing 1 to the [SHA\\_INT\\_CLEAR\\_REG](#) register.

7. Obtain the message digest:

- Read the message digest from registers [SHA\\_H\\_n\\_REG](#).

### **20.4.3 Message Digest**

After the hash computation completes, the SHA accelerator writes the message digest from the computation to registers [SHA\\_H\\_n\\_REG](#) ( $n$ : 0~15). The lengths of the generated message digest are different depending on different hash algorithms. For details, see Table 107 below:

**Table 107: The Storage and Length of Message digest from Different Algorithms**

Hash Algorithm	Length of Message Digest (in bits)	Storage <sup>1</sup>
SHA-1	160	SHA_H_0_REG ~ SHA_H_4_REG
SHA-224	224	SHA_H_0_REG ~ SHA_H_6_REG
SHA-256	256	SHA_H_0_REG ~ SHA_H_7_REG
SHA-384	384	SHA_H_0_REG ~ SHA_H_11_REG
SHA-512	512	SHA_H_0_REG ~ SHA_H_15_REG
SHA-512/224	224	SHA_H_0_REG ~ SHA_H_6_REG
SHA-512/256	256	SHA_H_0_REG ~ SHA_H_7_REG
SHA-512/ $t$ <sup>2</sup>	$t$	SHA_H_0_REG ~ SHA_H_ $x$ _REG

**Note:**

- The message digest are stored in registers from most significant bits to the least significant bits, with the first word stored in register [SHA\\_H\\_0\\_REG](#) and the second word stored in register [SHA\\_H\\_1\\_REG](#)... For details, please see subsection [20.4.1.2](#).
- The registers used for SHA-512/ $t$  algorithm depend on the value of  $t$ .  $x+1$  indicates the number of 32-bit registers used to store  $t$  bits of message digest, so that  $x = \text{roundup}(t/32)-1$ . For example:
  - When  $t = 8$ , then  $x = 0$ , indicating that the 8-bit long message digest is stored in the most significant 8 bits of register [SHA\\_H\\_0\\_REG](#);
  - When  $t = 32$ , then  $x = 0$ , indicating that the 32-bit long message digest is stored in register [SHA\\_H\\_0\\_REG](#);
  - When  $t = 132$ , then  $x = 4$ , indicating that the 132-bit long message digest is stored in registers [SHA\\_H\\_0\\_REG](#), [SHA\\_H\\_1\\_REG](#), [SHA\\_H\\_2\\_REG](#), [SHA\\_H\\_3\\_REG](#), and [SHA\\_H\\_4\\_REG](#).

### 20.4.4 Interrupt

SHA accelerator supports interrupt on the completion of computation when working in the DMA-SHA mode. To enable this function, write 1 to register [SHA\\_INT\\_ENA\\_REG](#). Note that the interrupt should be cleared by software after use via setting the [SHA\\_INT\\_CLEAR\\_REG](#) register to 1.

## 20.5 Base Address

Users can access SHA with two base addresses, which can be seen in Table 108. For more information about accessing peripherals from different buses, please see Chapter 1 *System and Memory*.

**Table 108: SHA Accelerator Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F43B000
PeriBUS2	0x6003B000

## 20.6 Register Summary

Name	Description	Address	Access
<b>Control/Status registers</b>			
<a href="#">SHA_CONTINUE_REG</a>	Continues SHA operation (only effective in Typical SHA mode)	0x0014	WO
<a href="#">SHA_BUSY_REG</a>	Indicates if SHA Accelerator is busy or not	0x0018	RO
<a href="#">SHA_DMA_START_REG</a>	Starts the SHA accelerator for DMA-SHA operation	0x001C	WO
<a href="#">SHA_START_REG</a>	Starts the SHA accelerator for Typical SHA operation	0x0010	WO
<a href="#">SHA_DMA_CONTINUE_REG</a>	Continues SHA operation (only effective in DMA-SHA mode)	0x0020	WO
<a href="#">SHA_INT_CLEAR_REG</a>	DMA-SHA interrupt clear register	0x0024	WO
<a href="#">SHA_INT_ENA_REG</a>	DMA-SHA interrupt enable register	0x0028	R/W
<b>Version Register</b>			
<a href="#">SHA_DATE_REG</a>	Version control register	0x002C	R/W
<b>Configuration Registers</b>			
<a href="#">SHA_MODE_REG</a>	Defines the algorithm of SHA accelerator	0x0000	R/W
<a href="#">SHA_T_STRING_REG</a>	String content register for calculating initial Hash Value (only effective for SHA-512/t)	0x0004	R/W
<a href="#">SHA_T_LENGTH_REG</a>	String length register for calculating initial Hash Value (only effective for SHA-512/t)	0x0008	R/W
<b>Memories</b>			
<a href="#">SHA_DMA_BLOCK_NUM_REG</a>	Block number register (only effective for DMA-SHA)	0x000C	R/W
<a href="#">SHA_H_0_REG</a>	Hash value	0x0040	R/W
<a href="#">SHA_H_1_REG</a>	Hash value	0x0044	R/W
<a href="#">SHA_H_2_REG</a>	Hash value	0x0048	R/W
<a href="#">SHA_H_3_REG</a>	Hash value	0x004C	R/W
<a href="#">SHA_H_4_REG</a>	Hash value	0x0050	R/W
<a href="#">SHA_H_5_REG</a>	Hash value	0x0054	R/W
<a href="#">SHA_H_6_REG</a>	Hash value	0x0058	R/W
<a href="#">SHA_H_7_REG</a>	Hash value	0x005C	R/W
<a href="#">SHA_H_8_REG</a>	Hash value	0x0060	R/W
<a href="#">SHA_H_9_REG</a>	Hash value	0x0064	R/W



Name	Description	Address	Access
SHA_H_10_REG	Hash value	0x0068	R/W
SHA_H_11_REG	Hash value	0x006C	R/W
SHA_H_12_REG	Hash value	0x0070	R/W
SHA_H_13_REG	Hash value	0x0074	R/W
SHA_H_14_REG	Hash value	0x0078	R/W
SHA_H_15_REG	Hash value	0x007C	R/W
SHA_M_0_REG	Message	0x0080	R/W
SHA_M_1_REG	Message	0x0084	R/W
SHA_M_2_REG	Message	0x0088	R/W
SHA_M_3_REG	Message	0x008C	R/W
SHA_M_4_REG	Message	0x0090	R/W
SHA_M_5_REG	Message	0x0094	R/W
SHA_M_6_REG	Message	0x0098	R/W
SHA_M_7_REG	Message	0x009C	R/W
SHA_M_8_REG	Message	0x00A0	R/W
SHA_M_9_REG	Message	0x00A4	R/W
SHA_M_10_REG	Message	0x00A8	R/W
SHA_M_11_REG	Message	0x00AC	R/W
SHA_M_12_REG	Message	0x00B0	R/W
SHA_M_13_REG	Message	0x00B4	R/W
SHA_M_14_REG	Message	0x00B8	R/W
SHA_M_15_REG	Message	0x00BC	R/W
SHA_M_16_REG	Message	0x00C0	R/W
SHA_M_17_REG	Message	0x00C4	R/W
SHA_M_18_REG	Message	0x00C8	R/W
SHA_M_19_REG	Message	0x00CC	R/W
SHA_M_20_REG	Message	0x00D0	R/W
SHA_M_21_REG	Message	0x00D4	R/W
SHA_M_22_REG	Message	0x00D8	R/W
SHA_M_23_REG	Message	0x00DC	R/W
SHA_M_24_REG	Message	0x00E0	R/W
SHA_M_25_REG	Message	0x00E4	R/W
SHA_M_26_REG	Message	0x00E8	R/W
SHA_M_27_REG	Message	0x00EC	R/W
SHA_M_28_REG	Message	0x00F0	R/W
SHA_M_29_REG	Message	0x00F4	R/W
SHA_M_30_REG	Message	0x00F8	R/W
SHA_M_31_REG	Message	0x00FC	R/W

## 20.7 Registers

Register 20.1: SHA\_START\_REG (0x0010)

(reserved)																															SHA_START				
31																															1	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**SHA\_START** Write 1 to start Typical SHA calculation. (WO)

Register 20.2: SHA\_CONTINUE\_REG (0x0014)

(reserved)																															SHA_CONTINUE		
31																															1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**SHA\_CONTINUE** Write 1 to continue Typical SHA calculation. (WO)

Register 20.3: SHA\_BUSY\_REG (0x0018)

(reserved)																															SHA_BUSY_STATE				
31																															1	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

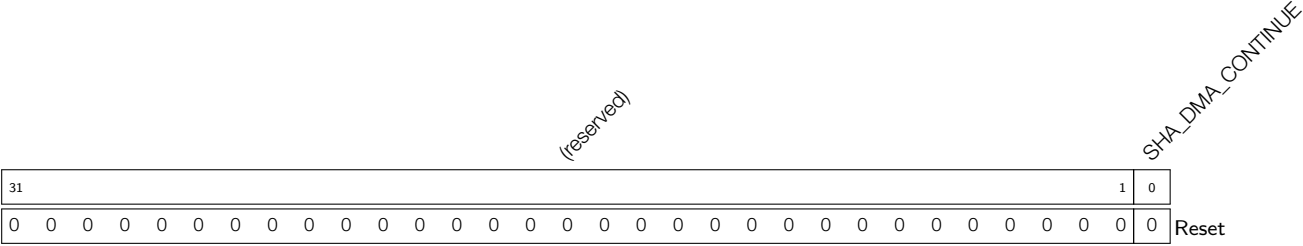
**SHA\_BUSY\_STATE** Indicates the states of SHA accelerator. (RO) 1'h0: idle 1'h1: busy

Register 20.4: SHA\_DMA\_START\_REG (0x001C)

(reserved)																															SHA_DMA_START																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

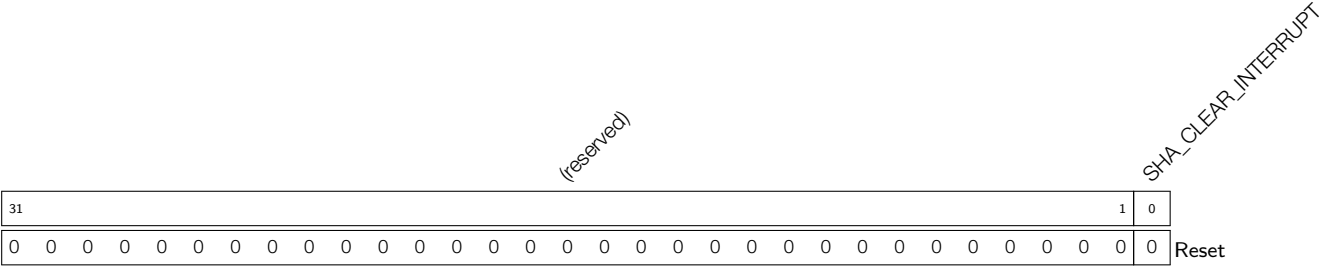
**SHA\_DMA\_START** Write 1 to start DMA-SHA calculation. (WO)

Register 20.5: SHA\_DMA\_CONTINUE\_REG (0x0020)



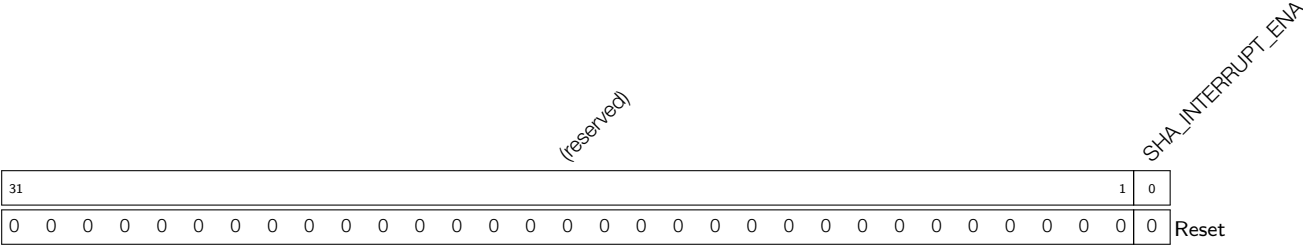
**SHA\_DMA\_CONTINUE** Write 1 to continue DMA-SHA calculation. (WO)

Register 20.6: SHA\_INT\_CLEAR\_REG (0x0024)



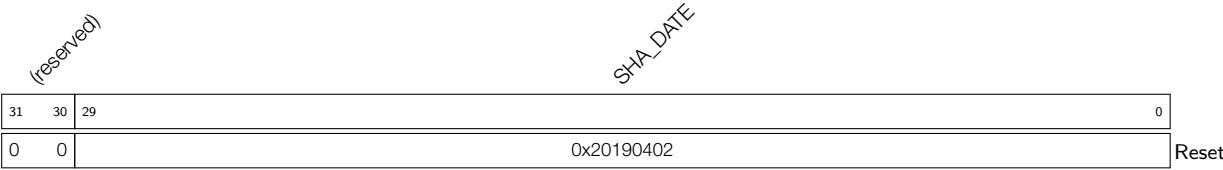
**SHA\_CLEAR\_INTERRUPT** Clears DMA-SHA interrupt. (WO)

Register 20.7: SHA\_INT\_ENA\_REG (0x0028)



**SHA\_INTERRUPT\_ENA** Enables DMA-SHA interrupt. (R/W)

Register 20.8: SHA\_DATE\_REG (0x002C)

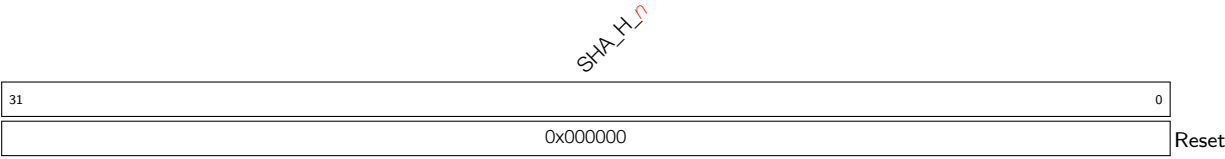


**SHA\_DATE** Version control register. (R/W)

Register 20.9: SHA\_MODE\_REG (0x0000)

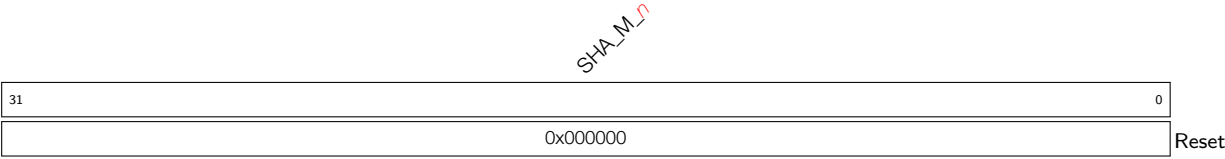
(reserved)																												SHA_MODE		
31																											3	2	0	
0 0																											0x0		Reset	

Register 20.13: SHA\_H\_n\_REG (n: 0-15) (0x0040+4\*n)



SHA\_H\_n Stores the nth 32-bit piece of the Hash value. (R/W)

Register 20.14: SHA\_M\_n\_REG (n: 0-31) (0x0080+4\*n)



SHA\_M\_n Stores the nth 32-bit piece of the message. (R/W)

## 21. RSA Accelerator

### 21.1 Introduction

The RSA Accelerator provides hardware support for high precision computation used in various RSA asymmetric cipher algorithms by significantly reducing their software complexity. Compared with RSA algorithms implemented solely in software, this hardware accelerator can speed up RSA algorithms significantly. Besides, the RSA Accelerator also supports operands of different lengths, which provides more flexibility during the computation.

### 21.2 Features

The following functionality is supported:

- Large-number modular exponentiation with two optional acceleration options
- Large-number modular multiplication
- Large-number multiplication
- Operands of different lengths
- Interrupt on completion of computation

### 21.3 Functional Description

The RSA Accelerator is activated by setting the DPORT\_PERIP\_CLK\_EN1\_REG bit in the DPORT\_CRYPT\_RSA\_CLK\_EN peripheral clock and clearing the DPORT\_RSA\_PD bit in the DPORT\_RSA\_PD\_CTRL\_REG register. This releases the RSA Accelerator from reset.

The RSA Accelerator is only available after the [RSA-related memories](#) are initialized. The content of the [RSA\\_CLEAN\\_REG](#) register is 0 during initialization and will become 1 after the initialization is done. Therefore, it is advised to wait until [RSA\\_CLEAN\\_REG](#) becomes 1 before using the RSA Accelerator.

The [RSA\\_INTERRUPT\\_ENA\\_REG](#) register is used to control the interrupt triggered on completion of computation. Write 1 or 0 to this register to enable or disable interrupt. By default, the interrupt function of the RSA Accelerator is enabled.

**Notice:**

ESP32-S2's [Digital Signature](#) module also calls the RSA accelerator. Therefore, users cannot access the RSA accelerator when [Digital Signature](#) is working.

### 21.3.1 Large Number Modular Exponentiation

Large-number modular exponentiation performs  $Z = X^Y \bmod M$ . The computation is based on Montgomery multiplication. Therefore, aside from the  $X$ ,  $Y$ , and  $M$  arguments, two additional ones are needed —  $\bar{r}$  and  $M'$ , which need to be calculated in advance by software.

RSA Accelerator supports operands of length  $N = 32 \times x$ , where  $x \in \{1, 2, 3, \dots, 128\}$ . The bit lengths of arguments  $Z$ ,  $X$ ,  $Y$ ,  $M$ , and  $\bar{r}$  can be arbitrary  $N$ , but all numbers in a calculation must be of the same length. The bit length of  $M'$  must be 32.

To represent the numbers used as operands, let us define a base- $b$  positional notation, as follows:

$$b = 2^{32}$$

Using this notation, each number is represented by a sequence of base- $b$  digits:

$$n = \frac{N}{32}$$

$$Z = (Z_{n-1}Z_{n-2} \cdots Z_0)_b$$

$$X = (X_{n-1}X_{n-2} \cdots X_0)_b$$

$$Y = (Y_{n-1}Y_{n-2} \cdots Y_0)_b$$

$$M = (M_{n-1}M_{n-2} \cdots M_0)_b$$

$$\bar{r} = (\bar{r}_{n-1}\bar{r}_{n-2} \cdots \bar{r}_0)_b$$

Each of the  $n$  values in  $Z_{n-1} \cdots Z_0$ ,  $X_{n-1} \cdots X_0$ ,  $Y_{n-1} \cdots Y_0$ ,  $M_{n-1} \cdots M_0$ ,  $\bar{r}_{n-1} \cdots \bar{r}_0$  represents one base- $b$  digit (a 32-bit word).

$Z_{n-1}$ ,  $X_{n-1}$ ,  $Y_{n-1}$ ,  $M_{n-1}$  and  $\bar{r}_{n-1}$  are the most significant bits of  $Z$ ,  $X$ ,  $Y$ ,  $M$ , while  $Z_0$ ,  $X_0$ ,  $Y_0$ ,  $M_0$  and  $\bar{r}_0$  are the least significant bits.

If we define  $R = b^n$ , the additional arguments can be calculated as  $\bar{r} = R^2 \bmod M$ , where  $R = b^n$ .

The following equation in the form compatible with the extended binary GCD algorithm can be written as

$$M^{-1} \times M + 1 = R \times R^{-1}$$

$$M' = M^{-1} \bmod b$$

Large-number modular exponentiation can be implemented as follows:

1. Write 1 or 0 to the [RSA\\_INTERRUPT\\_ENA\\_REG](#) register to enable or disable the interrupt function.
2. Configure relevant registers:
  - (a) Write  $(\frac{N}{32} - 1)$  to the [RSA\\_MODE\\_REG](#) register.
  - (b) Write  $M'$  to the [RSA\\_M\\_PRIME\\_REG](#) register.
  - (c) Configure registers related to the acceleration options, which are described later in Section 21.3.4.
3. Write  $X_i$ ,  $Y_i$ ,  $M_i$  and  $\bar{r}_i$  for  $i \in \{0, 1, \dots, n\}$  to memory blocks [RSA\\_X\\_MEM](#), [RSA\\_Y\\_MEM](#), [RSA\\_M\\_MEM](#) and [RSA\\_Z\\_MEM](#). The capacity of each memory block is 128 words. Each word of each memory block

can store one base- $b$  digit. The memory blocks use the little endian format for storage, i.e. the least significant digit of each number is in the lowest address.

Users need to write data to each memory block only according to the length of the number; data beyond this length are ignored.

4. Write 1 to the [RSA\\_MODEXP\\_START\\_REG](#) register to start computation.
5. Wait for the completion of computation, which happens when the content of [RSA\\_IDLE\\_REG](#) becomes 1 or the RSA interrupt occurs.
6. Read the result  $Z_i$  for  $i \in \{0, 1, \dots, n\}$  from [RSA\\_Z\\_MEM](#).
7. Write 1 to [RSA\\_CLEAR\\_INTERRUPT\\_REG](#) to clear the interrupt, if you have enabled the interrupt function.

After the computation, the [RSA\\_MODE\\_REG](#) register, memory blocks [RSA\\_Y\\_MEM](#) and [RSA\\_M\\_MEM](#), as well as the [RSA\\_M\\_PRIME\\_REG](#) remain unchanged. However,  $X_i$  in [RSA\\_X\\_MEM](#) and  $\bar{r}_i$  in [RSA\\_Z\\_MEM](#) computation are overwritten, and only these overwritten memory blocks need to be re-initialized before starting another computation.

### 21.3.2 Large Number Modular Multiplication

Large-number modular multiplication performs  $Z = X \times Y \bmod M$ . This computation is based on Montgomery multiplication. The same values  $\bar{r}$  and  $M'$  are derived by software.

The RSA Accelerator supports large-number modular multiplication with operands of 128 different lengths.

The computation can be executed as follows:

1. Write 1 or 0 to the [RSA\\_INTERRUPT\\_ENA\\_REG](#) register to enable or disable the interrupt function.
2. Configure relevant registers:
  - (a) Write  $(\frac{N}{32} - 1)$  to the [RSA\\_MODE\\_REG](#) register.
  - (b) Write  $M'$  to the [RSA\\_M\\_PRIME\\_REG](#) register.
3. Write  $X_i$ ,  $Y_i$ ,  $M_i$ , and  $\bar{r}_i$  for  $i \in \{0, 1, \dots, n\}$  to registers [RSA\\_X\\_MEM](#), [RSA\\_Y\\_MEM](#), [RSA\\_M\\_MEM](#) and [RSA\\_Z\\_MEM](#). The capacity of each memory block is 128 words. Each word of each memory block can store one base- $b$  digit. The memory blocks use the little endian format for storage, i.e. the least significant digit of each number is in the lowest address.

Users need to write data to each memory block only according to the length of the number; data beyond this length are ignored.
4. Write 1 to the [RSA\\_MODMULT\\_START\\_REG](#) register.
5. Wait for the completion of computation, which happens when the content of [RSA\\_IDLE\\_REG](#) becomes 1 or the RSA interrupt occurs.
6. Read the result  $Z_i$  for  $i \in \{0, 1, \dots, n\}$  from [RSA\\_Z\\_MEM](#).
7. Write 1 to [RSA\\_CLEAR\\_INTERRUPT\\_REG](#) to clear the interrupt, if you have enabled the interrupt function.

After the computation, the length of operands in [RSA\\_MODE\\_REG](#), the  $X_i$  in memory [RSA\\_X\\_MEM](#), the  $Y_i$  in memory [RSA\\_Y\\_MEM](#), the  $M_i$  in memory [RSA\\_M\\_MEM](#), and the  $M'$  in memory [RSA\\_M\\_PRIME\\_REG](#) remain unchanged. However, the  $\bar{r}_i$  in memory [RSA\\_Z\\_MEM](#) has already been overwritten, and only this overwritten memory block needs to be re-initialized before starting another computation.



### 21.3.3 Large Number Multiplication

Large-number multiplication performs  $Z = X \times Y$ . The length of result  $Z$  is twice that of operand  $X$  and operand  $Y$ . Therefore, the RSA Accelerator only supports Large Number Multiplication with operand length  $N = 32 \times x$ , where  $x \in \{0, 1, \dots, 64\}$ . The length  $\hat{N}$  of result  $Z$  is  $2 \times N$ .

The computation can be executed as follows:

1. Write 1 or 0 to the [RSA\\_INTERRUPT\\_ENA\\_REG](#) register to enable or disable the interrupt function.
2. Write  $(\frac{\hat{N}}{32} - 1)$ , i.e.  $(\frac{N}{16} - 1)$  to the [RSA\\_MODE\\_REG](#) register.
3. Write  $X_i$  and  $Y_i$  for  $i \in \{0, 1, \dots, n\}$  to registers [RSA\\_X\\_MEM](#) and [RSA\\_Z\\_MEM](#). The capacity of each memory block is 128 words. Each word of each memory block can store one base- $b$  digit. The memory blocks use the little endian format for storage, i.e. the least significant digit of each number is in the lowest address.  $n$  is  $\frac{N}{32}$ .

Write  $X_i$  for  $i \in \{0, 1, \dots, n\}$  to the address of the  $i$  words of the [RSA\\_X\\_MEM](#) register. Note that  $Y_i$  for  $i \in \{0, 1, \dots, n\}$  will not be written to the address of the  $i$  words of the [RSA\\_Z\\_MEM](#) register, but the address of the  $n + i$  words, i.e. the base address of the [RSA\\_Z\\_MEM](#) memory plus the address offset  $4 \times (n + i)$ .

Users need to write data to each memory block only according to the length of the number; data beyond this length are ignored.

4. Write 1 to the [RSA\\_MULT\\_START\\_REG](#) register.
5. Wait for the completion of computation, which happens when the content of [RSA\\_IDLE\\_REG](#) becomes 1 or the RSA interrupt occurs.
6. Read the result  $Z_i$  for  $i \in \{0, 1, \dots, n\}$  from the [RSA\\_Z\\_MEM](#) register.  $\hat{n}$  is  $2 \times n$ .
7. Write 1 to [RSA\\_CLEAR\\_INTERRUPT\\_REG](#) to clear the interrupt, if you have enabled the interrupt function.

After the computation, the length of operands in [RSA\\_MODE\\_REG](#) and the  $X_i$  in memory [RSA\\_X\\_MEM](#) remain unchanged. However, the  $Y_i$  in memory [RSA\\_Z\\_MEM](#) has already been overwritten, and only this overwritten memory block needs to be re-initialized before starting another computation.

### 21.3.4 Acceleration Options

ESP32-S2 RSA provides two acceleration options for the large-number modular exponentiation, which are SEARCH Option and the CONSTANT\_TIME Option. These two options are both disabled by default, but can be enabled at the same time.

When neither of these two options are enabled, the time required to calculate  $Z = X^Y \bmod M$  is solely determined by the lengths of operands. However, when either one of these two options is enabled, the time required is also correlated with the 0/1 distribution of  $Y$ .

To better illustrate the acceleration options, first assume  $Y$  is represented in binaries as

$$Y = (\tilde{Y}_{N-1}\tilde{Y}_{N-2}\cdots\tilde{Y}_{t+1}\tilde{Y}_t\tilde{Y}_{t-1}\cdots\tilde{Y}_0)_2$$

where,

- $N$  is the length of  $Y$ ,

- $\tilde{Y}_t$  is 1,
- $\tilde{Y}_{N-1}, \tilde{Y}_{N-2}, \dots, \tilde{Y}_{t+1}$  are all equal to 0,
- and  $\tilde{Y}_{t-1}, \tilde{Y}_{t-2}, \dots, \tilde{Y}_0$  are either 0 or 1 but exactly  $m$  bits should be equal to 0 and  $t-m$  bits 1, i.e. the Hamming weight of  $\tilde{Y}_{t-1}\tilde{Y}_{t-2}, \dots, \tilde{Y}_0$  is  $t-m$ .

When the acceleration options are enabled, the RSA accelerator:

- SEARCH Option
  - The accelerator ignores the bit positions of  $\tilde{Y}_i$ , where  $i > \alpha$ . Search position  $\alpha$  is set by configuring the [RSA\\_SEARCH\\_POS\\_REG](#) register. The maximum value of  $\alpha$  is  $N-1$ , which leads to the same result when this acceleration option is disabled. The best acceleration performance can be achieved by setting  $\alpha$  to  $t$ , in which case, all the  $\tilde{Y}_{N-1}, \tilde{Y}_{N-2}, \dots, \tilde{Y}_{t+1}$  of 0s are ignored during the calculation. Note that if you set  $\alpha$  to be less than  $t$ , then the result of the modular exponentiation  $Z = X^Y \bmod M$  will be incorrect.
- CONSTANT\_TIME Option
  - The accelerator speeds up the calculation by simplifying the calculation concerning the 0 bits of  $Y$ . Therefore, the higher the proportion of bits 0 against bits 1, the better the acceleration performance is.

We provide an example to demonstrate the performance of the RSA Accelerator when different acceleration options are enabled. Here we perform  $Z = X^Y \bmod M$  with  $N = 3072$  and  $Y = 65537$ . Table 110 below demonstrates the time costs when different acceleration options are enabled. It's obvious that the time cost can be dramatically reduced when acceleration option(s) is enabled. Here, we should also mention that,  $\alpha$  is set to 16 when the SEARCH option is enabled.

**Table 110: Acceleration Performance**

SEARCH Option	CONSTANT_TIME Option	Time Cost	Acceleration Performance by Percentage
Disabled	Disabled	376.405 ms	0%
Enabled	Disabled	2.260 ms	99.41%
Disabled	Enabled	1.203 ms	99.68%
Enabled	Enabled	1.165 ms	99.69%

## 21.4 Base Address

Users can access RSA with two base addresses, which can be seen in Table 111. For more information about accessing peripherals from different buses please see Chapter 1 *System and Memory*.

**Table 111: RSA Accelerator Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F43C000
PeriBUS2	0x6003C000

## 21.5 Memory Summary

Both the starting address and ending address in the following table are relative to RSA base addresses provided in Section 21.4.

**Table 112: RSA Accelerator Memory Blocks**

Name	Description	Size (byte)	Starting Address	Ending Address	Access
<a href="#">RSA_M_MEM</a>	Memory M	512	0x0000	0x01FF	WO
<a href="#">RSA_Z_MEM</a>	Memory Z	512	0x0200	0x03FF	R/W
<a href="#">RSA_Y_MEM</a>	Memory Y	512	0x0400	0x05FF	WO
<a href="#">RSA_X_MEM</a>	Memory X	512	0x0600	0x07FF	WO

## 21.6 Register Summary

The addresses in the following table are relative to RSA base addresses provided in Section 21.4.

Name	Description	Address	Access
<b>Configuration Registers</b>			
<a href="#">RSA_M_PRIME_REG</a>	Register to store M'	0x0800	R/W
<a href="#">RSA_MODE_REG</a>	RSA length mode	0x0804	R/W
<a href="#">RSA_CONSTANT_TIME_REG</a>	The constant_time option	0x0820	R/W
<a href="#">RSA_SEARCH_ENABLE_REG</a>	The search option	0x0824	R/W
<a href="#">RSA_SEARCH_POS_REG</a>	The search position	0x0828	R/W
<b>Status/Control Registers</b>			
<a href="#">RSA_CLEAN_REG</a>	RSA clean register	0x0808	RO
<a href="#">RSA_MODEXP_START_REG</a>	Modular exponentiation starting bit	0x080C	WO
<a href="#">RSA_MODMULT_START_REG</a>	Modular multiplication starting bit	0x0810	WO
<a href="#">RSA_MULT_START_REG</a>	Normal multiplication starting bit	0x0814	WO
<a href="#">RSA_IDLE_REG</a>	RSA idle register	0x0818	RO
<b>Interrupt Registers</b>			
<a href="#">RSA_CLEAR_INTERRUPT_REG</a>	RSA clear interrupt register	0x081C	WO
<a href="#">RSA_INTERRUPT_ENA_REG</a>	RSA interrupt enable register	0x082C	R/W
<b>Version Register</b>			
<a href="#">RSA_DATE_REG</a>	Version control register	0x0830	R/W

## 21.7 Registers

Register 21.1: RSA\_M\_PRIME\_REG (0x800)

31	0
0x00000000	
Reset	

**RSA\_M\_PRIME\_REG** Stores  $M'$ . (R/W)

Register 21.2: RSA\_MODE\_REG (0x804)

(reserved)															RSA_MODE																	
31															7	6	0															
0 0															0 0																Reset	

**RSA\_MODE** Stores the mode of modular exponentiation. (R/W)

Register 21.3: RSA\_CLEAN\_REG (0x808)

(reserved)																																RSA_CLEAN															
31																																1	0														
0																																0	0														
		Reset																																													

**RSA\_CLEAN** The content of this bit is 1 when memories complete initialization. (RO)

Register 21.4: RSA\_MODEXP\_START\_REG (0x80C)

(reserved)																																RSA_MODE									
31																																1	0								
0																																0	0								
		Reset																																							

**RSA\_MODEXP\_START** Set this bit to 1 to start the modular exponentiation. (WO)

**Register 21.5: RSA\_MODMULT\_START\_REG (0x0810)**

(reserved)																															RSA_MODMULT_START			
31																															1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**RSA\_MODMULT\_START** Set this bit to 1 to start the modular multiplication. (WO)

**Register 21.6: RSA\_MULT\_START\_REG (0x0814)**

(reserved)																														RSA_MULT_START	
31																														1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**RSA\_MULT\_START** Set this bit to 1 to start the multiplication. (WO)

**Register 21.7: RSA\_IDLE\_REG (0x0818)**

(reserved)																														RSA_IDLE	
31																														1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

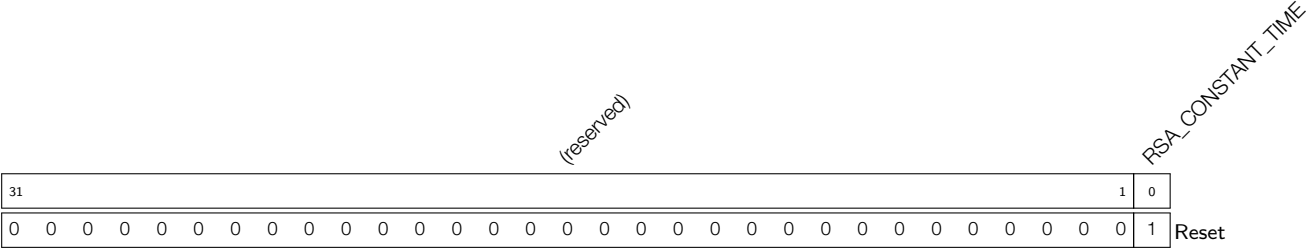
**RSA\_IDLE** The content of this bit is 1 when the RSA accelerator is idle. (RO)

**Register 21.8: RSA\_CLEAR\_INTERRUPT\_REG (0x081C)**

(reserved)																														RSA_CLEAR_INTERRUPT	
31																														1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

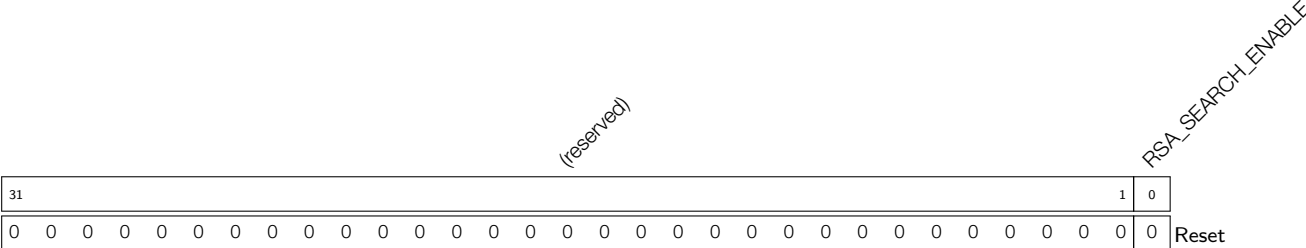
**RSA\_CLEAR\_INTERRUPT** Set this bit to 1 to clear the RSA interrupts. (WO)

Register 21.9: RSA\_CONSTANT\_TIME\_REG (0x0820)



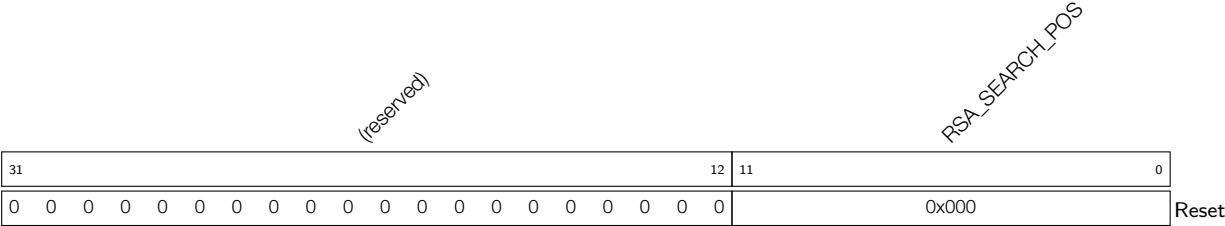
**RSA\_CONSTANT\_TIME\_REG** Set this bit to 0 to enable the acceleration option of constant\_time for modular exponentiation. Set to 1 to disable the acceleration (by default). (R/W)

Register 21.10: RSA\_SEARCH\_ENABLE\_REG (0x0824)



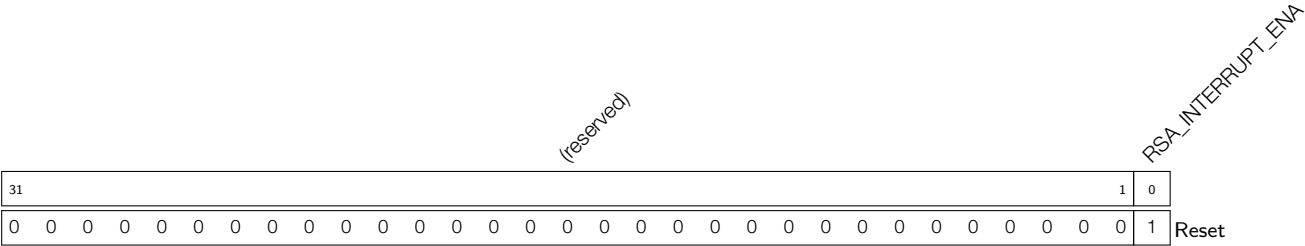
**RSA\_SEARCH\_ENABLE** Set this bit to 1 to enable the acceleration option of search for modular exponentiation. Set to 0 to disable the acceleration (by default). (R/W)

Register 21.11: RSA\_SEARCH\_POS\_REG (0x0828)



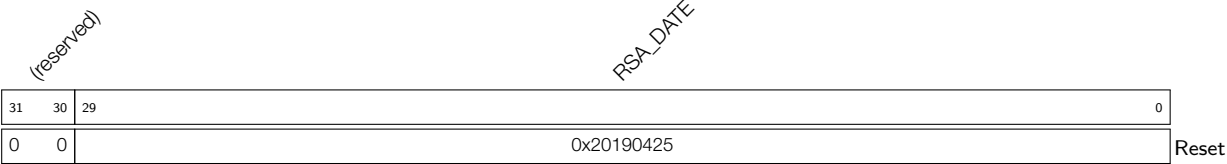
**RSA\_SEARCH\_POS** Is used to configure the starting address when the acceleration option of search is used. (R/W)

Register 21.12: RSA\_INTERRUPT\_ENA\_REG (0x082C)



**RSA\_INTERRUPT\_ENA** Set this bit to 1 to enable the RSA interrupt. This option is enabled by default.  
(R/W)

Register 21.13: RSA\_DATE\_REG (0x0830)



**RSA\_DATE** Version control register. (R/W)

## 22. Random Number Generator

### 22.1 Introduction

The ESP32-S2 contains a true random number generator, which generates random numbers that can be used for cryptographical operations, among other things.

### 22.2 Features

The random number generator generates true random numbers, which means no number generated within the specified range is more or less likely to appear than any other number.

### 22.3 Functional Description

When used correctly, every 32-bit value that the system reads from the [RNG\\_DATA\\_REG](#) register of the random number generator is a true random number. These true random numbers are generated based on the thermal noise in the system.

Either the high-speed ADC, SAR ADC, or both can be used as the noise source for the random number generator.

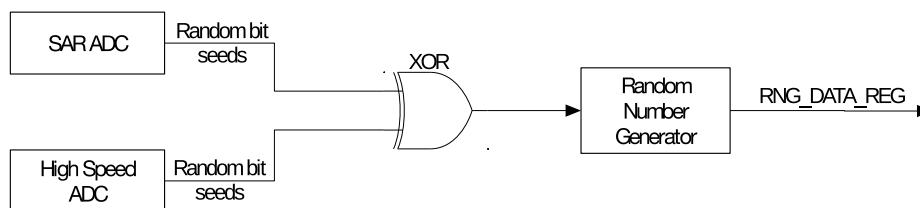


Figure 22-1. Noise Source

When there is noise coming from the high-speed ADC, the random number generator is fed with a 2-bit entropy in one APB clock cycle, which is normally 80 MHz. Thus, it is advisable to read the [RNG\\_DATA\\_REG](#) register at a maximum rate of 5 MHz to obtain the maximum entropy.

When there is noise coming from the SAR ADC, the random number generator is fed with a 2-bit entropy in one clock cycle of 8 MHz, which is generated from an internal RC oscillator (see [Reset and Clock](#) for details). Thus, it is advisable to read the [RNG\\_DATA\\_REG](#) register at a maximum rate of 500 kHz to obtain the maximum entropy.

A data sample of 2 GB, which is read from the random number generator at a rate of 5 MHz with only the high-speed ADC being enabled, has been tested using the Dieharder Random Number Testsuite (version 3.31.1). The sample passed all tests.

**Note:**

When the Wi-Fi module is enabled, the value read from the high-speed ADC can be saturated in some extreme cases, which lowers the entropy. Thus, it is advisable to enable the SAR ADC as the noise source for the random number generator when the Wi-Fi module is enabled.



To facilitate the generation of random numbers, a system API for generating random numbers will be provided. It's advisable to call this API directly.

## 22.4 Base Address

Users can access the random number generator with two base addresses, which can be seen in Table 114. For more information about accessing peripherals from different buses, please see Chapter 1 [System and Memory](#).

**Table 114: Random Number Generator Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F435000
PeriBUS2	0x60035000

## 22.5 Register Summary

The addresses in the following table are relative to the random number generator base addresses provided in Section 22.4.

Name	Description	Address	Access
<a href="#">RNG_DATA_REG</a>	Random number data	0x0110	RO

## 22.6 Register

The address in this section is relative to the random number generator base addresses provided in Section 22.4.

**Register 22.1: RNG\_DATA\_REG (0x0110)**

31	0
0x00000000	
Reset	

**RNG\_DATA** Random number source. (RO)

## 23. External Memory Encryption and Decryption

### 23.1 Overview

The ESP32-S2 SoC implements an External Memory Encryption and Decryption module that secures users' application code and data stored in the external memory (flash and external RAM). The encryption and decryption algorithm complies with the XTS-AES standard specified in [IEEE Std 1619-2007](#). Users can store proprietary firmware and sensitive data (for example credentials for gaining access to a private network) to the external flash, and general data to the external RAM.

### 23.2 Features

- General XTS-AES algorithm, compliant with IEEE Std 1619-2007
- Software-based manual encryption
- High-speed hardware auto encryption
- High-speed hardware auto decryption
- Encryption and decryption functions jointly determined by register configuration, eFuse parameters, and boot mode

### 23.3 Functional Description

The External Memory Encryption and Decryption module consists of three blocks, namely the Manual Encryption block, Auto Encryption block, and Auto Decryption block. The module architecture is shown in Figure 23-1.

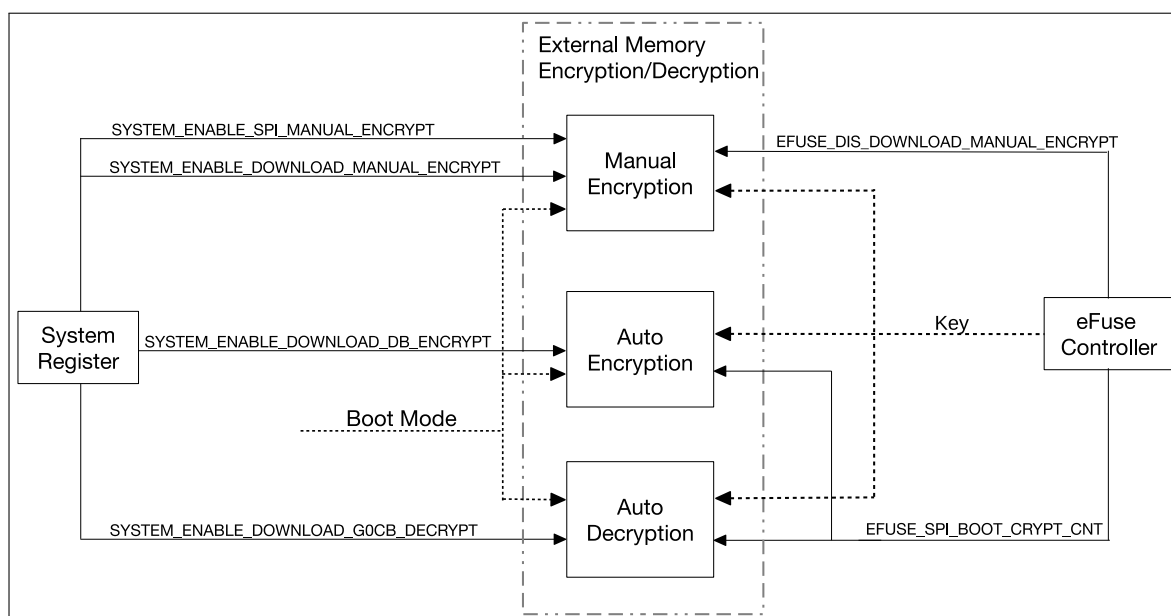


Figure 23-1. Architecture of the External Memory Encryption and Decryption Module

The Manual Encryption block can encrypt instructions and data which will then be written to the external flash as ciphertext through SPI1.

When the CPU writes the external RAM through cache, the Auto Encryption block automatically encrypts the data first, and the data is written to the external RAM as ciphertext.

When the CPU reads the external flash or RAM through cache, the Auto Decryption block automatically decrypts the ciphertext to retrieve instructions and data.

In the peripheral System Register, four bits in the SYSTEM\_EXTERNAL\_DEVICE\_ENCRYPT\_DECRYPT\_CONTROL\_REG register are relevant to external memory encryption and decryption:

- SYSTEM\_ENABLE\_DOWNLOAD\_MANUAL\_ENCRYPT
- SYSTEM\_ENABLE\_DOWNLOAD\_G0CB\_DECRYPT
- SYSTEM\_ENABLE\_DOWNLOAD\_DB\_ENCRYPT
- SYSTEM\_ENABLE\_SPI\_MANUAL\_ENCRYPT

The External Memory Encryption and Decryption module fetches two parameters from the peripheral eFuse Controller. These parameters are: EFUSE\_DIS\_DOWNLOAD\_MANUAL\_ENCRYPT and EFUSE\_SPI\_BOOT\_CRYPT\_CNT.

### 23.3.1 XTS Algorithm

The manual encryption, auto encryption, and auto decryption operations use the same algorithm, i.e., XTS algorithm. In real-life implementation, the XTS algorithm is characterized by “data unit” of 1024 bits. The “data unit” is defined in the *XTS-AES Tweakable Block Cipher* standard, section *XTS-AES encryption procedure*. More information on the XTS-AES algorithm can be found in [IEEE Std 1619-2007](#).

### 23.3.2 Key

The Manual Encryption block, Auto Encryption block, and Auto Decryption block share the same key to perform XTS algorithm. The key is provided by the eFuse hardware and protected from user access.

The key can be either 256 bits or 512 bits long. The key is determined by the content in one or two eFuse blocks from BLOCK4 ~ BLOCK9. For easy description, define:

- Block<sub>A</sub>, which refers to the block that has the key purpose set to EFUSE\_KEY\_PURPOSE\_XTS\_AES\_256\_KEY\_1. Block<sub>A</sub> contains 256-bit *Key<sub>A</sub>*.
- Block<sub>B</sub>, which refers to the block that has the key purpose set to EFUSE\_KEY\_PURPOSE\_XTS\_AES\_256\_KEY\_2. Block<sub>B</sub> contains 256-bit *Key<sub>B</sub>*.
- Block<sub>C</sub>, which refers to the block that has the key purpose set to EFUSE\_KEY\_PURPOSE\_XTS\_AES\_128\_KEY. Block<sub>C</sub> contains 256-bit *Key<sub>C</sub>*.

Table 116 shows how the *Key* is generated, depending on whether Block<sub>A</sub>, Block<sub>B</sub>, and Block<sub>C</sub> exists or not.

**Table 116: Key**

Block <sub>A</sub>	Block <sub>B</sub>	Block <sub>C</sub>	<i>Key</i>	<i>Key</i> Length (bit)
Yes	Yes	Don't care	<i>Key<sub>A</sub></i>    <i>Key<sub>B</sub></i>	512
Yes	No	Don't care	<i>Key<sub>A</sub></i>   0 <sup>256</sup>	512
No	Yes	Don't care	0 <sup>256</sup>    <i>Key<sub>B</sub></i>	512
No	No	Yes	<i>Key<sub>C</sub></i>	256

Block <sub>A</sub>	Block <sub>B</sub>	Block <sub>C</sub>	Key	Key Length (bit)
No	No	No	0 <sup>256</sup>	256

“Yes” indicates that the block exists; “No” indicates that the block does not exist; “0<sup>256</sup>” indicates a bit string that consists of 256-bit zeros; “||” is a bonding operator for joining one key string to another.

For more information on setting of key purposes, please refer to Chapter 16 *eFuse Controller*.

### 23.3.3 Target Memory Space

The target memory space refers to a continuous address space in the external memory where the encrypted result is stored. The target memory space can be uniquely determined by three relevant parameters: *type*, *size*, and *base\_addr*. They are defined as follows:

- *type*: the type of the external memory, either flash or external RAM. Value 0 indicates flash, 1 indicates external RAM.
- *size*: the size of the target memory space, in unit of bytes. One single encryption operation supports either 16, 32, or 64 bytes of data.
- *base\_addr*: the base address of the target memory space. It is a physical address aligned to *size*, i.e.,  $base\_addr \% size == 0$ .

Assume encrypted 16 bytes written to address 0x130 ~ 0x13F in the external flash, then, the target memory space is 0x130 ~ 0x13F, *type* is 0 (flash), *size* is 16 (bytes), and *base\_addr* 0x130.

The encryption of any length (must be multiples of 16 bytes) of data can be completed separately in multiple operations. Each operation can have individual target memory space and the relevant parameters.

For auto encryption and auto decryption, these parameters are automatically defined by hardware. For manual encryption, these parameters should be configured manually by users.

### 23.3.4 Data Padding

For auto encryption and auto decryption, data padding is automatically completed by hardware. For manual encryption, data padding should be completed manually by users. The Manual Encryption block is equipped with 16 registers, i.e., XTS\_AES\_PLAIN\_*n*\_REG (*n*: 0-15), that are dedicated to data padding and can store up to 512 bits of plaintext at a time.

Actually, the Manual Encryption block does not care where the plaintext comes from, but only where the ciphertext is to be stored. Because of the strict correspondence between plaintext and ciphertext, in order to better describe how the plaintext is stored in the register heap, it is assumed that the plaintext is stored in the target memory space in the first place and replaced by ciphertext after encryption. Therefore, the following description no longer has the concept of “plaintext”, but uses “target memory space” instead. However, users should note that the plaintext can come from anywhere, and that they should understand how the plaintext is stored in the register heap.

#### How mapping works between target memory space and registers:

Assume a word is stored in *address*, define  $offset = address \% 64$ ,  $n = \frac{offset}{4}$ , then the word will be stored in register XTS\_AES\_PLAIN\_*n*\_REG.

For example, if the *size* of the target memory space is 64, then all the 16 registers will be used for data storage. The mapping between *offset* and registers is shown in Table 117.

**Table 117: Mapping Between Offsets and Registers**

<i>offset</i>	Register	<i>offset</i>	Register
0x00	<a href="#">XTS_AES_PLAIN_0_REG</a>	0x20	<a href="#">XTS_AES_PLAIN_8_REG</a>
0x04	<a href="#">XTS_AES_PLAIN_1_REG</a>	0x24	<a href="#">XTS_AES_PLAIN_9_REG</a>
0x08	<a href="#">XTS_AES_PLAIN_2_REG</a>	0x28	<a href="#">XTS_AES_PLAIN_10_REG</a>
0x0C	<a href="#">XTS_AES_PLAIN_3_REG</a>	0x2C	<a href="#">XTS_AES_PLAIN_11_REG</a>
0x10	<a href="#">XTS_AES_PLAIN_4_REG</a>	0x30	<a href="#">XTS_AES_PLAIN_12_REG</a>
0x14	<a href="#">XTS_AES_PLAIN_5_REG</a>	0x34	<a href="#">XTS_AES_PLAIN_13_REG</a>
0x18	<a href="#">XTS_AES_PLAIN_6_REG</a>	0x38	<a href="#">XTS_AES_PLAIN_14_REG</a>
0x1C	<a href="#">XTS_AES_PLAIN_7_REG</a>	0x3C	<a href="#">XTS_AES_PLAIN_15_REG</a>

### 23.3.5 Manual Encryption Block

The Manual Encryption block is a peripheral module. It is equipped with registers that can be accessed by the CPU directly. Registers embedded in this block, System registers, eFuse parameters, and boot mode jointly configure and control this block. Please note that currently the Manual Encryption block can only encrypt flash.

The manual encryption requires software participation. The steps are as follows:

1. Configure XTS\_AES:

- Set [XTS\\_AES\\_DESTINATION\\_REG](#) register to  $type = 0$ .
- Set [XTS\\_AES\\_PHYSICAL\\_ADDRESS\\_REG](#) register to  $base\_addr$ .
- Set [XTS\\_AES\\_LINESIZE\\_REG](#) register to  $\frac{size}{32}$ .

For definitions of  $type$ ,  $base\_addr$ ,  $size$ , please refer to Section 23.3.3.

2. Fill registers [XTS\\_AES\\_PLAIN\\_n\\_REG](#) ( $n$ : 0-15) in with plaintext (refer to Section 23.3.4). Registers that are not used can be written into any value.
3. Poll [XTS\\_AES\\_STATE\\_REG](#) until it reads 0 that indicates the Manual Encryption block is idle.
4. Activate encryption by writing 1 to [XTS\\_AES\\_TRIGGER\\_REG](#) register.
5. Wait for the encryption to complete. Poll register [XTS\\_AES\\_STATE\\_REG](#) until it reads 2.  
Steps 1 ~ 5 complete the encryption operation, where  $Key$  is used.
6. Grant SPI1 access to the encrypted result by writing 1 to [XTS\\_AES\\_RELEASE\\_REG](#) register.  
[XTS\\_AES\\_STATE\\_REG](#) will read 3 afterwards.
7. Call SPI1 and write the encrypted result to the external flash.
8. Destroy the encrypted result by writing 1 to [XTS\\_AES\\_DESTROY\\_REG](#). [XTS\\_AES\\_STATE\\_REG](#) register will read 0 afterwards.

Repeat the steps above to complete multiple encryption operations.

**The Manual Encryption block is operational only with granted permission.** The operating conditions are:

- In SPI Boot mode  
If bit [SYSTEM\\_ENABLE\\_SPI\\_MANUAL\\_ENCRYPT](#) in register

SYSTEM\_EXTERNAL\_DEVICE\_ENCRYPT\_DECRYPT\_CONTROL\_REG is 1, the Manual Encryption block is granted permission. Otherwise, it is not operational.

- In Download Boot mode

If bit [SYSTEM\\_ENABLE\\_DOWNLOAD\\_MANUAL\\_ENCRYPT](#) in register

SYSTEM\_EXTERNAL\_DEVICE\_ENCRYPT\_DECRYPT\_CONTROL\_REG is 1 and the eFuse parameter [EFUSE\\_DIS\\_DOWNLOAD\\_MANUAL\\_ENCRYPT](#) is 0, the Manual Encryption block is granted permission.

Otherwise, it is not operational.

**Note:**

- Even though the CPU can skip cache and get the encrypted result directly by reading the external memory, software can by no means access *Key*.
- The Manual Encryption block needs to call the AES accelerator to perform encryption. Therefore, users cannot access AES accelerator during the process.

### 23.3.6 Auto Encryption Block

The Auto Encryption block is not a conventional peripheral, and is not equipped with registers. Therefore, the CPU cannot directly access this block. The System Register, eFuse parameters, and boot mode jointly control this block.

**The Auto Encryption block is operational only with granted permission.** The operating conditions are:

- In SPI Boot mode

If the 3-bit parameter SPI\_BOOT\_CRYPT\_CNT has 1 or 3 bits set to 1, then the Auto Encryption block is granted permission. Otherwise, it is not operational.

- In Download Boot mode

If bit [SYSTEM\\_ENABLE\\_DOWNLOAD\\_DB\\_ENCRYPT](#) in register

SYSTEM\_EXTERNAL\_DEVICE\_ENCRYPT\_DECRYPT\_CONTROL\_REG is 1, the Auto Encryption block is granted permission. Otherwise, it is not operational.

**Note:**

- When the Auto Encryption block is operational, the CPU will read data from the external RAM via the cache. The Auto Encryption block automatically encrypts the data and write it to the external RAM. The entire encryption process does not need software participation and is transparent to the cache. Software cannot access the encryption *Key*.
- When the Auto Encryption block is not operational, it will ignore the CPU's request to access cache and do not process the data. Therefore, data will be written to the external RAM as plaintext.

### 23.3.7 Auto Decryption Block

The Auto Decryption block is not a conventional peripheral, and is not equipped with registers. Therefore, the CPU cannot directly access this block. The System Register, eFuse parameters, and boot mode jointly control and configure this block.

**The Auto Decryption block is operational only with granted permission.** The operating conditions are:

- In SPI Boot mode

If the 3-bit parameter SPI\_BOOT\_CRYPT\_CNT has 1 or 3 bits set to 1, then the Auto Decryption block is granted permission. Otherwise, it is not operational.

- In Download Boot mode

If bit [SYSTEM\\_ENABLE\\_DOWNLOAD\\_G0CB\\_DECRYPT](#) in register

SYSTEM\_EXTERNAL\_DEVICE\_ENCRYPT\_DECRYPT\_CONTROL\_REG is 1, the Auto Decryption block is granted permission. Otherwise, it is not operational.

**Note:**

- When the Auto Decryption block is operational, the CPU will read instructions and data from the external memory via cache. The Auto Decryption block automatically decrypts and retrieves the instructions and data. The entire decryption process does not need software participation and is transparent to the cache. Software cannot access the decryption *Key*.
- When the Auto Decryption block is not operational, it does not have any effect on the contents stored in the external memory, be they encrypted or unencrypted. What the CPU reads via cache is the original information stored in the external memory.

## 23.4 Base Address

Users can access the Manual Encryption block with two base addresses, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter 1: [System and Memory](#).

**Table 118: Manual Encryption Block Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F43A000
PeriBUS2	0x6003A000

## 23.5 Register Summary

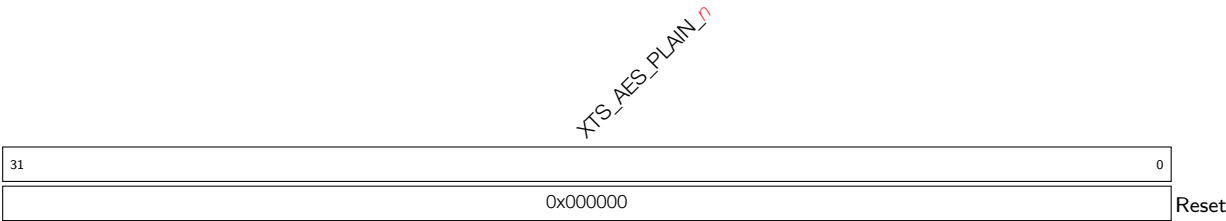
The addresses in the following table are relative to the Manual Encryption block's base addresses provided in Section 23.4.

Name	Description	Address	Access
<b>Plaintext Register Heap</b>			
<a href="#">XTS_AES_PLAIN_0_REG</a>	Plaintext register 0	0x0100	R/W
<a href="#">XTS_AES_PLAIN_1_REG</a>	Plaintext register 1	0x0104	R/W
<a href="#">XTS_AES_PLAIN_2_REG</a>	Plaintext register 2	0x0108	R/W
<a href="#">XTS_AES_PLAIN_3_REG</a>	Plaintext register 3	0x010C	R/W
<a href="#">XTS_AES_PLAIN_4_REG</a>	Plaintext register 4	0x0110	R/W
<a href="#">XTS_AES_PLAIN_5_REG</a>	Plaintext register 5	0x0114	R/W
<a href="#">XTS_AES_PLAIN_6_REG</a>	Plaintext register 6	0x0118	R/W
<a href="#">XTS_AES_PLAIN_7_REG</a>	Plaintext register 7	0x011C	R/W
<a href="#">XTS_AES_PLAIN_8_REG</a>	Plaintext register 8	0x0120	R/W
<a href="#">XTS_AES_PLAIN_9_REG</a>	Plaintext register 9	0x0124	R/W
<a href="#">XTS_AES_PLAIN_10_REG</a>	Plaintext register 10	0x0128	R/W

Name	Description	Address	Access
<a href="#">XTS_AES_PLAIN_11_REG</a>	Plaintext register 11	0x012C	R/W
<a href="#">XTS_AES_PLAIN_12_REG</a>	Plaintext register 12	0x0130	R/W
<a href="#">XTS_AES_PLAIN_13_REG</a>	Plaintext register 13	0x0134	R/W
<a href="#">XTS_AES_PLAIN_14_REG</a>	Plaintext register 14	0x0138	R/W
<a href="#">XTS_AES_PLAIN_15_REG</a>	Plaintext register 15	0x013C	R/W
<b>Configuration Registers</b>			
<a href="#">XTS_AES_LINESIZE_REG</a>	Configures the size of target memory space	0x0140	R/W
<a href="#">XTS_AES_DESTINATION_REG</a>	Configures the type of the external memory	0x0144	R/W
<a href="#">XTS_AES_PHYSICAL_ADDRESS_REG</a>	Physical address	0x0148	R/W
<b>Control/Status Registers</b>			
<a href="#">XTS_AES_TRIGGER_REG</a>	Activates AES algorithm	0x014C	WO
<a href="#">XTS_AES_RELEASE_REG</a>	Release control	0x0150	WO
<a href="#">XTS_AES_DESTROY_REG</a>	Destroys control	0x0154	WO
<a href="#">XTS_AES_STATE_REG</a>	Status register	0x0158	RO
<b>Version Register</b>			
<a href="#">XTS_AES_DATE_REG</a>	Version control register	0x015C	RO

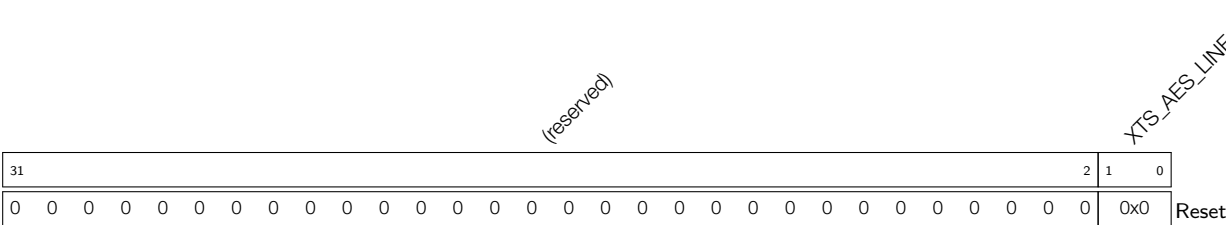
23.6 Registers

Register 23.1: XTS\_AES\_PLAIN\_n\_REG (n: 0-15) (0x0100+4\*n)



**XTS\_AES\_PLAIN\_n** This register stores *n*th 32-bit piece of plaintext. (R/W)

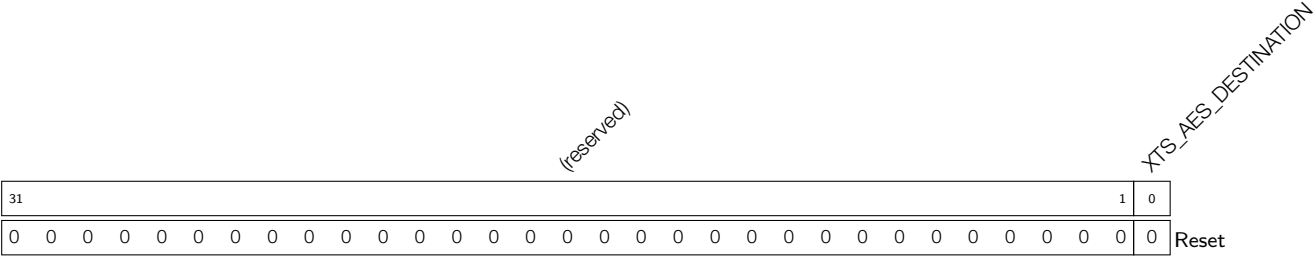
Register 23.2: XTS\_AES\_LINESIZE\_REG (0x0140)



**XTS\_AES\_LINESIZE** Configures the data size of a single encryption. 0: 128 bits; 1: 256 bits; 2: 512 bits. (R/W)

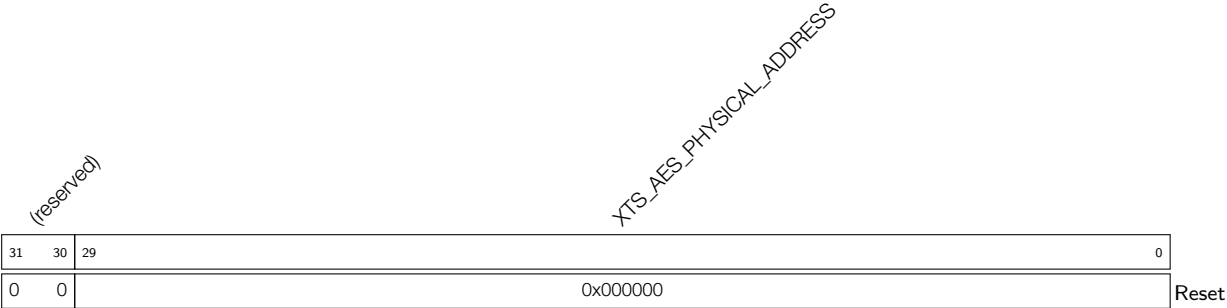


Register 23.3: XTS\_AES\_DESTINATION\_REG (0x0144)



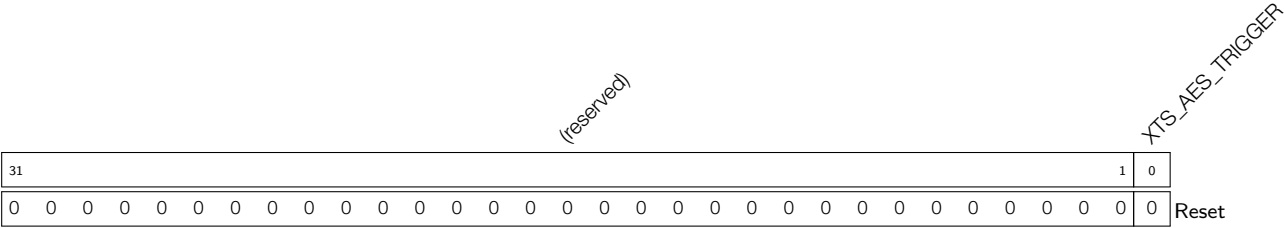
**XTS\_AES\_DESTINATION** Configures the type of the external memory. Currently, it must be set to 0, as the Manual Encryption block only supports flash encryption. Errors may occur if users write 1. 0: flash; 1: external RAM. (R/W)

Register 23.4: XTS\_AES\_PHYSICAL\_ADDRESS\_REG (0x0148)



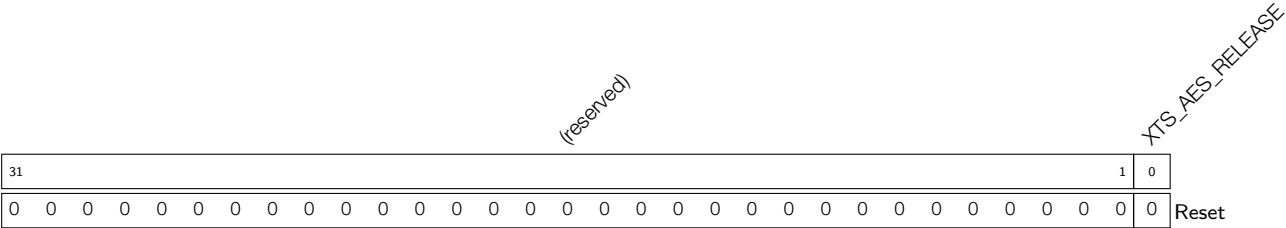
**XTS\_AES\_PHYSICAL\_ADDRESS** Physical address. (R/W)

Register 23.5: XTS\_AES\_TRIGGER\_REG (0x014C)



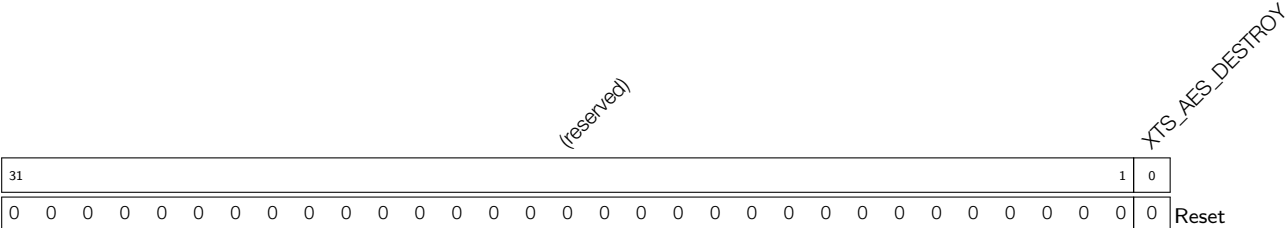
**XTS\_AES\_TRIGGER** Set to enable manual encryption. (WO)

Register 23.6: XTS\_AES\_RELEASE\_REG (0x0150)



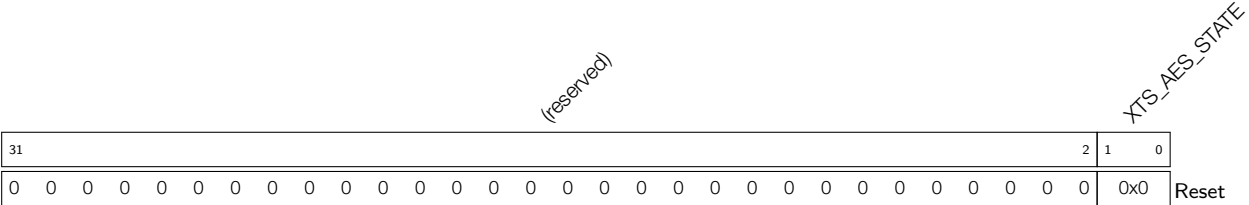
**XTS\_AES\_RELEASE** Set to grant SPI1 access to encrypted result. (WO)

Register 23.7: XTS\_AES\_DESTROY\_REG (0x0154)



**XTS\_AES\_DESTROY** Set to destroy encrypted result. (WO)

Register 23.8: XTS\_AES\_STATE\_REG (0x0158)



**XTS\_AES\_STATE** Indicates the status of the Manual Encryption block. (RO)

- 0x0 (XTS\_AES\_IDLE): idle;
- 0x1 (XTS\_AES\_BUSY): busy with encryption;
- 0x2 (XTS\_AES\_DONE): encryption is completed, but the encrypted result is not accessible to SPI;
- 0x3 (XTS\_AES\_RELEASE): encrypted result is accessible to SPI.

Register 23.9: XTS\_AES\_DATE\_REG (0x015C)

(reserved)		XTS_AES_DATE	
31	30	29	0
0	0	0x20190514	
		Reset	

**XTS\_AES\_DATE** Version control register. (RO)

## 24. Permission Control

### 24.1 Overview

In ESP32-S2, different type of buses and modules can have different permissions to access the memory. The permission control module is used to manage all the access permissions through corresponding permission control registers.

### 24.2 Features

- Permission controls for the CPU buses to access the internal memory
  - SRAM partially adopts unified permission control and partially adopts split permission control.
  - RTC FAST SRAM adopts split permission control.
  - RTC SLOW SRAM adopts split permission control.
- Access to the external memory is managed by both the memory management unit (MMU) and the permission control module.

### 24.3 Functional Description

#### 24.3.1 Internal Memory Permission Controls

The internal memory resources in ESP32-S2 include ROM, SRAM, RTC FAST Memory, and RTC SLOW Memory. The breakdown of each memory is as follows:

- The total ROM size is 128 KB and consists of two 64 KB blocks.
- The total SRAM size is 320 KB and consists of four 8 KB blocks and eighteen 16 KB blocks, numbered Block 0 ~ 21. The offset address range of each block is shown in Table 120. For the base addresses accessed by different buses please refer to Section 24.3.1.1, 24.3.1.2, and 24.3.1.3. Each of the four 8 KB blocks (Block 0 ~ 3) is equipped with an independent permission control register. The eighteen 16 KB blocks (Block 4 ~ 21) are regarded as a large storage space as a whole, which is managed in split parts (i.e., the storage space is split into high and low areas with separate permission controls). The split address cannot fall within the address range of Block 4 ~ 5.
- RTC FAST Memory adopts split permission control. The high and low areas have independent permission control registers.
- RTC SLOW Memory also adopts split permission control. The high and low areas have independent permission control registers.

The following sections describe the buses and modules in ESP32-S2 that can access the above-mentioned internal memory resources.

**Table 120: Offset Address Range of Each SRAM Block**

SRAM Block	Offset Start Address	Offset End Address
Block 0	0x0000	0x1FFF
Block 1	0x2000	0x3FFF
Block 2	0x4000	0x5FFF

SRAM Block	Offset Start Address	Offset End Address
Block 3	0x6000	0x7FFF
Block 4	0x8000	0xBFFF
Block 5	0xC000	0xFFFF
Block 6	0x10000	0x13FFF
Block 7	0x14000	0x17FFF
Block 8	0x18000	0x1BFFF
Block 9	0x1C000	0x1FFFF
Block 10	0x20000	0x23FFF
Block 11	0x24000	0x27FFF
Block 12	0x28000	0x2BFFF
Block 13	0x2C000	0x2FFFF
Block 14	0x30000	0x33FFF
Block 15	0x34000	0x37FFF
Block 16	0x38000	0x3BFFF
Block 17	0x3C000	0x3FFFF
Block 18	0x40000	0x43FFF
Block 19	0x44000	0x47FFF
Block 20	0x48000	0x4BFFF
Block 21	0x4C000	0x4FFFF

### 24.3.1.1 Permission Control for the Instruction Bus (IBUS)

The address range on IBUS that requires permission control is 0x4002\_0000 ~ 0x4007\_1FFF, where SRAM memory blocks and RTC FAST Memory are located. Access to different memory blocks is controlled by independent permission control registers which are configured by software.

The permission control registers related to IBUS are also controlled by the [PMS\\_PRO\\_IRAM0\\_LOCK](#) signal. When this signal is set to 1, the values configured in the permission control registers will be locked and cannot be changed. At the same time, the value of the [PMS\\_PRO\\_IRAM0\\_LOCK](#) signal will also remain at 1 and cannot be changed. The value of the [PMS\\_PRO\\_IRAM0\\_LOCK](#) signal is reset to 0 only when the CPU is reset.

#### SRAM Memory Blocks

Through IBUS, the CPU can Read (data) from SRAM, Write (data) to SRAM, and eXecute (an instruction), indicated as R/W/X, respectively. The base address of SRAM accessed by the CPU is 0x4002\_0000. Software can configure valid access types to each SRAM block in advance. The configuration of the permission control registers is shown in Table 121.

**Table 121: Permission Control for IBUS to Access SRAM**

Register	Bit Positions	Description
<a href="#">PMS_PRO_IRAM0_1_REG</a>	[11:9]	Configure the permission for IBUS to access SRAM Block 3 (from high to low as W/R/X)
	[8:6]	Configure the permission for IBUS to access SRAM Block 2 (from high to low as W/R/X)

Register	Bit Positions	Description
	[5:3]	Configure the permission for IBUS to access SRAM Block 1 (from high to low as W/R/X)
	[2:0]	Configure the permission for IBUS to access SRAM Block 0 (from high to low as W/R/X)
<a href="#">PMS_PRO_IRAM0_2_REG</a>	[22:20]	Configure the permission for IBUS to access the high address range in SRAM Block 4 ~ 21 (from high to low as W/R/X)
	[19:17]	Configure the permission for IBUS to access the low address range in SRAM Block 4 ~ 21 (from high to low as W/R/X)
	[16:0]*	Configure the split address of IBUS in the SRAM Block 4 ~ 21 address range. The base address is 0x4000_0000. See below for instructions on how to configure split address.

**Note:****\* Configuring split address:**

Split address is 32 bits in width and equal to base address plus the configured field value multiplied by 4. For example, if 0x10000 is written to [PMS\\_PRO\\_IRAM0\\_2\\_REG](#) [16:0], the split address of IBUS in the SRAM Block 4 ~ 21 address range is equal to 0x4004\_0000. Note that the split address on IBUS and DBUS0 cannot fall within the address range of Block 0 ~ 5.

If the type of access initiated by the CPU to SRAM through IBUS does not match the configured access type, the permission control module will directly deny this access. For example, if software allows the IBUS bus to only read and write SRAM Block5, but the CPU initiates an instruction fetch, the permission control module will deny this fetch access.

An access denial does not block the access initiated by the CPU, which means:

- The write operation will not take effect and will not change the data in the memory block.
- Read and fetch operations will get invalid data.

If the interrupt that indicates an illegitimate access to SRAM via IBUS is enabled, the permission control module will record the current access address and access type, and give an interrupt signal at the same time. If the access is denied repeatedly, the hardware only records the first error message. The access error interrupt is a level signal and needs to be cleared by software. When the interrupt is cleared, the related error record is also cleared at the same time.

**RTC FAST Memory**

The CPU can perform R/W/X access to the RTC FAST Memory through IBUS. As RTC FAST Memory adopts split permission control, software needs to configure access permissions for the high and low address ranges, respectively. The configuration of the permission control registers is shown in Table 122.

**Table 122: Permission Control for IBUS to Access RTC FAST Memory**

Register	Bit Positions	Description
<a href="#">PMS_PRO_IRAM0_3_REG</a>	[16:14]	Configure the permission for IBUS to access the high address range in RTC FAST Memory (from high to low as W/R/X)

Register	Bit Positions	Description
	[13:11]	Configure the permission for IBUS to access the low address range in RTC FAST Memory (from high to low as W/R/X)
	[10:0]	Configure the user-configurable split address of IBUS in the RTC FAST address range. The base address is 0x4007_0000. For how to configure the user-configurable split address, please refer to the instructions above.

If the type of access initiated by the CPU to RTC Fast Memory through IBUS does not match the configured access type, the permission control module will directly deny this access. If the interrupt that indicates an illegitimate access to the RTC Fast Memory via IBUS is enabled, the permission control module will record the current access address and access type, and give an interrupt signal at the same time. If the access is denied repeatedly, the hardware only records the first error message. The access error interrupt is a level signal and needs to be cleared by software. When the interrupt is cleared, the related error record is also cleared at the same time.

### 24.3.1.2 Permission Control for the Data Bus (DBUS0)

The CPU can access two address ranges through DBUS0: 0x3FFB\_0000 ~ 0x3FFF\_FFFF and 0x3FF9\_E000 ~ 0x3FF9\_FFFF, where SRAM memory blocks and RTC FAST Memory are located. Access to different memory blocks is controlled by independent permission control registers which are configured by software.

The permission control registers related to DBUS0 are also controlled by the [PMS\\_PRO\\_DRAM0\\_LOCK](#) signal. When this signal is set to 1, the configured values of the permission control registers will be locked and cannot be changed. The value of the [PMS\\_PRO\\_DRAM0\\_LOCK](#) signal will also remain at 1 and cannot be changed. The value of the [PMS\\_PRO\\_DRAM0\\_LOCK](#) signal is reset to 0 only when the CPU is reset.

#### SRAM Memory Blocks

The CPU can perform R/W access to SRAM via DBUS0, with the based address 0x3FFB\_0000. Software can configure allowed access types initiated by DBUS0 to each SRAM block in advance. The configuration of the permission control registers is shown in Table 123.

**Table 123: Permission Control for DBUS0 to Access SRAM**

Register	Bit Positions	Description
<a href="#">PMS_PRO_DRAM0_1_REG</a>	[28:27]	Configure the permission for DBUS0 to access the high address range in SRAM Block 4 ~ 21 (from high to low as W/R)
	[26:25]	Configure the permission for DBUS0 to access the low address range in SRAM Block 4 ~ 21 (from high to low as W/R)
	[24:8]	Configure the user-configurable split address of DBUS0 in the SRAM Block 4 ~ 21 address range. The base address is 0x3FFB_0000. For how to configure the user-configurable split address, please refer to the instructions in Section 24.3.1.1.
	[7:6]	Configure the permission for DBUS0 to access SRAM Block3 (from high to low as W/R)
	[5:4]	Configure the permission for DBUS0 to access SRAM Block2 (from high to low as W/R)

Register	Bit Positions	Description
	[3:2]	Configure the permission for DBUS0 to access SRAM Block1 (from high to low as W/R)
	[1:0]	Configure the permission for DBUS0 to access SRAM Block0 (from high to low as W/R)

If the type of access initiated by the CPU to SRAM through DBUS0 does not match the configured access type, the permission control module will directly deny this access. If the interrupt that indicates an illegitimate access to SRAM via DBUS0 is enabled, the permission control module will record the current access address, access type, and access size (in bytes, halfwords, or words), and give an interrupt signal at the same time. If the access is denied repeatedly, the hardware only records the first error message. The access error interrupt is a level signal and needs to be cleared by software. When the interrupt is cleared, the related error record is also cleared at the same time.

### RTC FAST Memory

The CPU can perform R/W access to RTC FAST Memory through DBUS0. As RTC FAST Memory adopts split permission control, software needs to configure access permissions for the high and low address ranges, respectively. The configuration of the permission control registers is shown in Table 124.

**Table 124: Permission Control for DBUS0 to Access RTC FAST Memory**

Register	Bit Positions	Description
PMS_PRO_DRAM0_2_REG	[14:13]	Configure the permission for DBUS0 to access the high address range in RTC FAST Memory (from high to low as W/R)
	[12:11]	Configure the permission for DBUS0 to access the low address range in RTC FAST Memory (from high to low as W/R)
	[10:0]	Configure the user-configurable split address of DBUS0 within the RTC FAST Memory address range. The base address is 0x3FF9_E000. For how to configure the user-configurable split address, please refer to the instructions in Section 24.3.1.1.

If the type of access initiated by the CPU to RTC FAST Memory through DBUS0 does not match the configured access type, the permission control module will directly deny this access. If the interrupt that indicates an illegitimate access to RTC FAST Memory via DBUS0 is enabled, the permission control module will record the current access address, access type, and access size (in bytes, halfwords, or words), and give an interrupt signal at the same time. If the access is denied repeatedly, the hardware only records the first error message. The access error interrupt is a level signal and needs to be cleared by software. When the interrupt is cleared, the related error record is also cleared at the same time.

#### 24.3.1.3 Permission Control for On-chip DMA

The on-chip DMA can access the internal memory through Internal DMA, Copy DMA RX (receiving) channel, or Copy DMA TX (transmitting) channel, using the address range 0x3FFB\_0000 ~ 0x3FFF\_FFFF (0x3FFB\_0000 is the base address). Software can configure the type of DMA access to each SRAM block in advance. The configuration of the permission control registers is shown in Table 125, where **XX** can be APB, TX, or RX,



corresponding to the configuration registers of Internal DMA, TX Copy DMA, and RX Copy DMA, respectively. More details about DMA can be found in Chapter 7 [DMA Controller](#).

The permission control registers related to DMA are also controlled by the PMS\_DMA\_XX\_I\_LOCK signal. When this signal is set to 1, the configured values of the permission control registers will be locked and cannot be changed. The value of the PMS\_DMA\_XX\_I\_LOCK signal will also remain at 1 and cannot be changed. The PMS\_DMA\_XX\_I\_LOCK signal value is reset to 0 only when the CPU is reset.

**Table 125: Permission Control for On-chip DMA to Access SRAM**

Register	Bit Positions	Description
PMS_DMA_XX_I_1_REG	[28:27]	Configure the permission for DMA to access the high address range in SRAM Block 4 ~ 21 (from high to low as W/R)
	[26:25]	Configure the permission for DMA to access the low address range in SRAM Block 4 ~ 21 (from high to low as W/R)
	[24:8]	Configure the user-configurable split address of DMA in SRAM Block 4 ~ 21 address range. The base address is 0x3FFB_0000. For how to configure the user-configurable split address, please refer to the instructions in Section 24.3.1.1.
	[7:6]	Configure the permission for DMA to access SRAM Block3 (from high to low as W/R)
	[5:4]	Configure the permission for DMA to access SRAM Block2 (from high to low as W/R)
	[3:2]	Configure the permission for DMA to access SRAM Block1 (from high to low as W/R)
	[1:0]	Configure the permission for DMA to access SRAM Block0 (from high to low as W/R)

If the type of access initiated by the on-chip DMA to SRAM does not match the configured access type, the permission control module will directly deny this access. If the interrupt that indicates an illegitimate access to SRAM via Internal DMA, TX Copy DMA, or RX Copy DMA is enabled, the permission control module will record the current access address and access type, and give an interrupt signal at the same time. If the access is denied repeatedly, the hardware only records the first error message. The access error interrupt is a level signal and needs to be cleared by software. When the interrupt is cleared, the related error record is also cleared at the same time.

#### 24.3.1.4 Permission Control for PeriBus1

The CPU can access the address range 0x3F40\_0000 ~ 0x3F4F\_FFFF through PeriBus1, where RTC SLOW SRAM is located. PeriBus1 can perform R/W operations. The read/write permission is controlled by software.

Software can also configure whether PeriBus1 is allowed to access peripherals that are within the address range 0x3F40\_0000 ~ 0x3F4B\_FFFF.

As the CPU can initiate predictive read operations through PeriBus1, it may cause peripheral FIFO read errors. To avoid FIFO read errors, the FIFO addresses of the corresponding peripherals have already been written in hardware and can no longer be changed, as Table 126 shows. In addition, four software-configurable address

registers PMS\_PRO\_DPORT\_2 ~ 5\_REG are reserved for users to write into addresses read-protected from PeriBus1 reading. The configuration of the permission control registers is shown in Table 127.

**Table 126: Peripherals and FIFO Address**

Peripherals	FIFO Address
ADDR_RTCSLOW	0x6002_1000
ADDR_FIFO_UART0	0x6000_0000
ADDR_FIFO_UART1	0x6001_0000
ADDR_FIFO_UART2	0x6002_E000
ADDR_FIFO_I2S0	0x6000_F004
ADDR_FIFO_I2S1	0x6002_D004
ADDR_FIFO_RMT_CH0	0x6001_6000
ADDR_FIFO_RMT_CH1	0x6001_6004
ADDR_FIFO_RMT_CH2	0x6001_6008
ADDR_FIFO_RMT_CH3	0x6001_600C
ADDR_FIFO_I2C_EXT0	0x6001_301C
ADDR_FIFO_I2C_EXT1	0x6002_701C
ADDR_FIFO_USB_0	0x6008_0020
ADDR_FIFO_USB_1_L	0x6008_1000
ADDR_FIFO_USB_1_H	0x6009_0FFF

The permission control registers related to PeriBus1 are also controlled by the [PMS\\_PRO\\_DPORT\\_LOCK](#) signal. When this signal is set to 1, the configured values of the permission registers will be locked and cannot be changed. The value of the [PMS\\_PRO\\_DPORT\\_LOCK](#) signal will also remain at 1 and cannot be changed. The value of the [PMS\\_PRO\\_DPORT\\_LOCK](#) signal is reset to 0 only when the CPU is reset.

**Table 127: Permission Control for PeriBus1**

Register	Bit Positions	Description
<a href="#">PMS_PRO_DPORT_1_REG</a>	[19:16]	The value of each bit determines whether to enable the corresponding reserved FIFO address.
	[15:14]	Configure the permission for PeriBus1 to access the high address range in RTC SLOW Memory (from high to low as W/R)
	[13:12]	Configure the permission for PeriBus1 to access the low address range in RTC SLOW Memory (from high to low as W/R)
	[11:1]	Configure the user-configurable split address of PeriBus1 in the RTC SLOW Memory address range. The base address is 0x3F42_1000. For how to configure the user-configurable split address, please refer to the instructions in Section 24.3.1.1.
	[0]	Configure whether PeriBus1 can access the peripherals within the address range 0x3F40_0000 ~ 0x3F4B_FFFF
<a href="#">PMS_PRO_DPORT_2_REG</a>	[17:0]	The zeroth address unreadable to PeriBus1, enabled by <a href="#">PMS_PRO_DPORT_RESERVE_FIFO_VALID</a> [16]
<a href="#">PMS_PRO_DPORT_3_REG</a>	[17:0]	The first address unreadable to PeriBus1, enabled by <a href="#">PMS_PRO_DPORT_RESERVE_FIFO_VALID</a> [17]

Register	Bit Positions	Description
<a href="#">PMS_PRO_DPORT_4_REG</a>	[17:0]	The second address unreadable to PeriBus1, enabled by <a href="#">PMS_PRO_DPORT_RESERVE_FIFO_VALID</a> [18]
<a href="#">PMS_PRO_DPORT_5_REG</a>	[17:0]	The third address unreadable to PeriBus1, enabled by <a href="#">PMS_PRO_DPORT_RESERVE_FIFO_VALID</a> [19]

If PeriBus1 initiates a read access to unreadable address, or if the type of the access to RTC SLOW Memory does not match the configured type, the permission control module will directly deny this access. If the interrupt that indicates an illegitimate access to RTC SLOW Memory by PeriBus1 is enabled, the permission control module will record the current access address, access type, and access size (in bytes, halfwords, or words), and give an interrupt signal at the same time. If the access is denied repeatedly, the hardware only records the first error message. The access error interrupt is a level signal and needs to be cleared by software. When the interrupt is cleared, the related error record is also cleared at the same time.

### 24.3.1.5 Permission Control for PeriBus2

The two address ranges on PeriBus2 that require permission control are 0x5000\_0000 ~ 0x5000\_1FFF and 0x6000\_0000 ~ 0x600B\_FFFF where RTC SLOW Memory and peripheral modules are located.

PeriBus2 supports R/W/X access to RTC SLOW Memory through two address ranges, namely RTCSlow\_0 and RTCSlow\_1. Software can configure the allowed access types. The configuration of the permission control registers is shown in Table 128.

PeriBus2 does not support fetch operations on peripherals. If a fetch operation is initiated on a peripheral, invalid data will be obtained.

The permission control registers related to PeriBus2 are also controlled by the [PMS\\_PRO\\_AHB\\_LOCK](#) signal. When this signal is configured to 1, the values of the permission control registers are locked and cannot be changed. The value of the [PMS\\_PRO\\_AHB\\_LOCK](#) signal will also remain at 1 and cannot be changed. The value of the [PMS\\_PRO\\_AHB\\_LOCK](#) signal is reset to 0 only when the CPU is reset.

**Table 128: Permission Control for PeriBus2 to Access RTC SLOW Memory**

Register	Bit Positions	Description
<a href="#">PMS_PRO_AHB_1_REG</a>	[16:14]	Configure the permission for PeriBus2 to access the high address range in RTCSlow_0 (from high to low as W/R/X)
	[13:11]	Configure the permission for PeriBus2 to access the low address range in RTCSlow_0 (from high to low as W/R/X)
	[10:0]	Configure the user-configurable split address of PeriBus2 in the RTCSlow_0 address range. The base address is 0x5000_1000. For how to configure the user-configurable split address, please refer to the instructions in Section 24.3.1.1.
<a href="#">PMS_PRO_AHB_2_REG</a>	[16:14]	Configure the permission for PeriBus2 to access the high address range in RTCSlow_1 (from high to low as W/R/X)
	[13:11]	Configure the permission for PeriBus2 to access the low address range in RTCSlow_1 (from high to low as W/R/X)

Register	Bit Positions	Description
	[10:0]	Configure the user-configurable split address of PeriBus2 in the RTCSlow_1 address range. The base address is 0x6002_1000. For how to configure the user-configurable split address, please refer to the instructions in Section 24.3.1.1.

If the type of the access to RTCSlow\_0 or RTCSlow\_1 made by the CPU through PeriBus2 does not match the configured type, the permission control module will directly deny this access. If the interrupt that indicates an illegitimate access by PeriBus2 is enabled, the permission control module will record the current access address and access type, and give an interrupt signal at the same time. If the access is denied repeatedly, the hardware only records the first error message. The access error interrupt is a level signal and needs to be cleared by software. When the interrupt is cleared, the related error record is also cleared at the same time.

### 24.3.1.6 Permission Control for Cache

Among the SRAM blocks, only Block 0 ~ 3 can be assigned to Icache and Dcache. Icache or Dcache can access up to 16 KB internal memory, which means, they can access up to two blocks. The 16 KB address range accessed by Icache or Dcache is divided into two ranges: high address range (indicated by “\_H” and low address range (indicated by “\_L”).

The user can allocate Block 0 ~ 3 to Icache\_H, Icache\_L, Dcache\_H, or Dcache\_L by configuring the field [PMS\\_PRO\\_CACHE\\_CONNECT](#) in register [PMS\\_PRO\\_CACHE\\_1\\_REG](#). The detailed configuration is shown in Table 129, where *FIELD* indicates field [PMS\\_PRO\\_CACHE\\_CONNECT](#).

**Table 129: Configuration of Register [PMS\\_PRO\\_CACHE\\_1\\_REG](#)**

SRAM Block 0-3	Dcache_H	Dcache_L	Icache_H	Icache_L
Block 0	<i>FIELD</i> [3]	<i>FIELD</i> [2]	<i>FIELD</i> [1]	<i>FIELD</i> [0]
Block 1	<i>FIELD</i> [7]	<i>FIELD</i> [6]	<i>FIELD</i> [5]	<i>FIELD</i> [4]
Block 2	<i>FIELD</i> [11]	<i>FIELD</i> [10]	<i>FIELD</i> [9]	<i>FIELD</i> [8]
Block 3	<i>FIELD</i> [15]	<i>FIELD</i> [14]	<i>FIELD</i> [13]	<i>FIELD</i> [12]

#### Notes:

- One block can be assigned to only one of Dcache\_H, Dcache\_L, Icache\_H, and Icache\_L, which means that one bit at most can be set to 1 in the same row in Table 129.
- One of Dcache\_H, Dcache\_L, Icache\_H, and Icache\_L can occupy only one block, which means that one bit at most can be set to 1 in the same column in Table 129.
- SRAM memory occupied by the cache cannot be accessed by the CPU. Non-occupied SRAM memory can still be accessed by the CPU.

### 24.3.1.7 Permission Control of Other Types of Internal Memory

In ESP32-S2, software can read Trace memory to get information about the runtime status of the CPU. Additionally, it can also be used as a debug channel to communicate with software running on the CPU. Software can enable the Trace memory function by configuring register [PMS\\_PRO\\_TRACE\\_1](#). This register is also controlled by the [PMS\\_PRO\\_TRACE\\_LOCK](#) signal. When this signal is set to 1, the value configured in the

register will be locked and cannot be changed. At the same time, the value of the [PMS\\_PRO\\_TRACE\\_LOCK](#) signal will also remain at 1 and cannot be changed. The value of the [PMS\\_PRO\\_TRACE\\_LOCK](#) signal is reset to 0 only when the CPU is reset.

After the Trace memory function is enabled, register PMS\_OCCUPY\_3 needs to be configured to select one block from SRAM Block 4 ~ 21 as Trace memory. Only one block can be used as Trace memory, in which case, it can no longer be accessed by the CPU. Register PMS\_OCCUPY\_3 is controlled by the [PMS\\_OCCUPY\\_LOCK](#) signal. When this signal is configured as 1, the value configured in the register will be locked and cannot be changed. At the same time, the value of the [PMS\\_OCCUPY\\_LOCK](#) signal will also remain at 1 and cannot be changed. The [PMS\\_OCCUPY\\_LOCK](#) signal value is reset to 0 the CPU is reset.

When an SRAM block is selected as Trace Memory, neither IBUS, DBUS, nor DMA can access this block.

### 24.3.2 External Memory Permission Control

The CPU can access the external flash and SRAM by SPI1, EDMA or cache. The former two support direct access, and the cache supports indirect access.

#### 24.3.2.1 Cache MMU

The Cache MMU is used to control the permission for cache and EDMA to access the external memory. Its primary function is to convert virtual addresses to physical addresses. The MMU needs to be configured before enabling cache and EDMA. When a cache miss occurs or the data is written back to the external memory, the cache controller will automatically access the MMU and generate a physical address to access the external memory. When EDMA reads and writes the external SRAM, the EDMA controller also automatically accesses the MMU and generates a physical address to access the external SRAM.

**Table 130: MMU Entries**

MMU Entries	Bit Positions	Description
SRAM	[16]	The external memory attribute indicates whether cache/EDMA accesses the external flash or SRAM. If bit 15 is set, cache/EDMA accesses flash. If bit 16 is set, cache/EDMA accesses SRAM. The two bits cannot be set at the same time.
Flash	[15]	
Invalid	[14]	Cleared if MMU entry is valid.
Page number	[13:0]	For MMU purposes, external memory is divided in memory blocks called 'pages'. The page size is fixed at 64 KB. The page number of the physical address space indicates which page is accessed by cache/EDMA.

When cache or EDMA accesses an invalid MMU page or the external memory attribute is not specified in the MMU, an MMU error interrupt will be triggered.

#### 24.3.2.2 External Memory Permission Controls

The hardware divides the physical address of the external memory (flash + SRAM) into eight (4 + 4) areas. Each area can be individually configured for W/R/X access. Software needs to configure the size of each area and its access type in advance.

The eight areas correspond to eight sets of registers, and the flash and SRAM each has four sets. Each set of registers contains three parts as follows:

1. Attribute list: APB\_CTRL\_X\_ACS\_n\_ATTR\_REG has 3 bits in total, which are W/R/X bits from high to low;
2. Area start address: APB\_CTRL\_X\_ACS\_n\_ADDR\_REG, which represents the physical address;
3. Area length: APB\_CTRL\_X\_ACS\_n\_SIZE\_REG in multiples of 64 KB;

Wherein “X” represents flash or SRAM, “n” represents 0 ~ 3. Note that the total size of the four areas corresponding to flash must be 1 GB, so is the total size of the four areas corresponding to SRAM.

When the CPU directly accesses external memory, the permission control module only monitors the CPU's write requests, not the read requests. The control module checks whether the access type matches the configured type according to the accessed physical address. If the area does not allow write access, the permission module directly rejects write access, records the information about the current access, and triggers an access error interrupt.

Cache miss, cache write back, and cache prefetch operations will trigger W/R/X requests from the cache to the external memory. The permission control module will query for the access attributes of the area according to the accessed physical address, and compare the the cache request against the pre-configured attributes. Only when the two match, the permission control module will pass the request to the external memory. If the cache initiates an R/F request, the permission control module will send the access attributes of the area to the cache for storage. When the access request is inconsistent with the current access attributes, the hardware records the information about the current access and triggers an access error interrupt.

When the CPU accesses the cache, the permission control module must check the attributes of the CPU access by comparing it with the local access attribute list. Only if the two match, the CPU access is allowed. When the access request is inconsistent with the current access attributes, the hardware records the information about the current access and triggers an access error interrupt.

If the access attribute check fails repeatedly, the hardware only records the first error message, and the subsequent error messages will be discarded directly. The access error interrupt is a level signal. When software clears the access error interrupt flag, the error record will be cleared at the same time.

### 24.3.3 Non-Aligned Access Permission Control

ESP32-S2 supports monitoring of non-word-aligned access. When PeriBus1 or PeriBus2 initiates a non-word address alignment or non-word sized access to a peripheral device, an interrupt or system exception may be triggered. The following table lists all possible access types and their corresponding results, where *IN* means interrupt, *EX* means exceptions, and  $\checkmark$  means no action is triggered.

**Table 131: Non-Aligned Access to Peripherals**

Accessed by	Access Address	Access Unit	Read	Write
PeriBus2	0XXXXX_XXX0	byte	IN	IN
		halfword	IN	IN
		word	✓	✓
	0XXXXX_XXX1	byte	IN	IN
		halfword	✓	IN
		word	✓	IN
	0XXXXX_XXX2	byte	IN	IN
		halfword	IN	IN
		word	✓	IN
Address range 0x3F40_0000 - 0x3F4B_FFFF	0XXXXX_XXX0	byte	IN	IN
		halfword	IN	IN
		word	✓	✓
	0XXXXX_XXX1	byte	IN	IN
		halfword	EX	EX
		word	EX	EX
	0XXXXX_XXX2	byte	IN	IN
		halfword	IN	IN
		word	EX	EX
Address range 0x3F4C_0000 - 0x3F4F_FFFF	0XXXXX_XXX0	byte	✓	✓
		halfword	✓	✓
		word	✓	✓
	0XXXXX_XXX1	byte	IN	IN
		halfword	EX	EX
		word	EX	EX
	0XXXXX_XXX2	byte	IN	IN
		halfword	IN	IN
		word	EX	EX

## 24.4 Base Address

Users can access the permission control module with the base address as shown in Table 132. For more information about accessing peripherals please see Chapter 1: [System and Memory](#).

**Table 132: Permission Control Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F4C1000

## 24.5 Register Summary

The address in the following table represents the address offset (relative address) with the respect to the peripheral base address, not the absolute address. For detailed information about the permission control base address, please refer to Section 24.4.



Name	Description	Address	Access
<b>Control Registers</b>			
<a href="#">PMS_PRO_IRAM0_0_REG</a>	IBUS permission control register 0.	0x0010	R/W
<a href="#">PMS_PRO_DRAM0_0_REG</a>	DBUS permission control register 0.	0x0028	R/W
<a href="#">PMS_PRO_DPORT_0_REG</a>	PeriBus1 permission control register 0.	0x003C	R/W
<a href="#">PMS_PRO_AHB_0_REG</a>	PeriBus2 permission control register 0.	0x005C	R/W
<a href="#">PMS_PRO_TRACE_0_REG</a>	Trace memory permission control register 0.	0x0070	R/W
<a href="#">PMS_PRO_CACHE_0_REG</a>	Cache permission control register 0.	0x0078	R/W
<a href="#">PMS_DMA_APB_I_0_REG</a>	Internal DMA permission control register 0.	0x008C	R/W
<a href="#">PMS_DMA_RX_I_0_REG</a>	RX Copy DMA permission control register 0.	0x009C	R/W
<a href="#">PMS_DMA_TX_I_0_REG</a>	TX Copy DMA permission control register 0.	0x00AC	R/W
<a href="#">PMS_CACHE_SOURCE_0_REG</a>	Cache access permission control register 0.	0x00C4	R/W
<a href="#">PMS_APB_PERIPHERAL_0_REG</a>	Peripheral access permission control register 0.	0x00CC	R/W
<a href="#">PMS_OCCUPY_0_REG</a>	Occupy permission control register 0.	0x00D4	R/W
<a href="#">PMS_CACHE_TAG_ACCESS_0_REG</a>	Cache tag permission control register 0.	0x00E4	R/W
<a href="#">PMS_CACHE_MMU_ACCESS_0_REG</a>	Cache MMU permission control register 0.	0x00EC	R/W
<a href="#">PMS_CLOCK_GATE_REG_REG</a>	Clock gate register of permission control.	0x0104	R/W
<b>Configuration Registers</b>			
<a href="#">PMS_PRO_IRAM0_1_REG</a>	IBUS permission control register 1.	0x0014	R/W
<a href="#">PMS_PRO_IRAM0_2_REG</a>	IBUS permission control register 2.	0x0018	R/W
<a href="#">PMS_PRO_IRAM0_3_REG</a>	IBUS permission control register 3.	0x001C	R/W
<a href="#">PMS_PRO_DRAM0_1_REG</a>	DBUS permission control register 1.	0x002C	R/W
<a href="#">PMS_PRO_DRAM0_2_REG</a>	DBUS permission control register 2.	0x0030	R/W
<a href="#">PMS_PRO_DPORT_1_REG</a>	PeriBus1 permission control register 1.	0x0040	R/W
<a href="#">PMS_PRO_DPORT_2_REG</a>	PeriBus1 permission control register 2.	0x0044	R/W
<a href="#">PMS_PRO_DPORT_3_REG</a>	PeriBus1 permission control register 3.	0x0048	R/W
<a href="#">PMS_PRO_DPORT_4_REG</a>	PeriBus1 permission control register 4.	0x004C	R/W
<a href="#">PMS_PRO_DPORT_5_REG</a>	PeriBus1 permission control register 5.	0x0050	R/W
<a href="#">PMS_PRO_AHB_1_REG</a>	PeriBus2 permission control register 1.	0x0060	R/W
<a href="#">PMS_PRO_AHB_2_REG</a>	PeriBus2 permission control register 2.	0x0064	R/W
<a href="#">PMS_PRO_TRACE_1_REG</a>	Trace memory permission control register 1.	0x0074	R/W
<a href="#">PMS_PRO_CACHE_1_REG</a>	Cache permission control register 1.	0x007C	R/W
<a href="#">PMS_DMA_APB_I_1_REG</a>	Internal DMA permission control register 1.	0x0090	R/W
<a href="#">PMS_DMA_RX_I_1_REG</a>	RX Copy DMA permission control register 1.	0x00A0	R/W
<a href="#">PMS_DMA_TX_I_1_REG</a>	TX Copy DMA permission control register 1.	0x00B0	R/W
<a href="#">PMS_APB_PERIPHERAL_1_REG</a>	Peripheral access permission control register 1.	0x00D0	R/W
<a href="#">PMS_OCCUPY_1_REG</a>	Occupy permission control register 1.	0x00D8	R/W
<a href="#">PMS_OCCUPY_3_REG</a>	Occupy permission control register 3.	0x00E0	R/W
<a href="#">PMS_CACHE_TAG_ACCESS_1_REG</a>	Cache tag permission control register 1.	0x00E8	R/W
<a href="#">PMS_CACHE_MMU_ACCESS_1_REG</a>	Cache MMU permission control register 1.	0x00F0	R/W
<b>Interrupt Registers</b>			
<a href="#">PMS_PRO_IRAM0_4_REG</a>	IBUS permission control register 4.	0x0020	varies
<a href="#">PMS_PRO_IRAM0_5_REG</a>	IBUS status register.	0x0024	RO
<a href="#">PMS_PRO_DRAM0_3_REG</a>	DBUS permission control register 3.	0x0034	varies



Name	Description	Address	Access
PMS_PRO_DRAM0_4_REG	DBus status register.	0x0038	RO
PMS_PRO_DPORT_6_REG	PeriBus1 permission control register 6.	0x0054	varies
PMS_PRO_DPORT_7_REG	PeriBus1 status register.	0x0058	RO
PMS_PRO_AHB_3_REG	PeriBus2 permission control register 3.	0x0068	varies
PMS_PRO_AHB_4_REG	PeriBus2 status register.	0x006C	RO
PMS_PRO_CACHE_2_REG	Cache permission control register 2.	0x0080	varies
PMS_PRO_CACHE_3_REG	Lcache status register.	0x0084	RO
PMS_PRO_CACHE_4_REG	Dcache status register.	0x0088	RO
PMS_DMA_APB_I_2_REG	Internal DMA permission control register 2.	0x0094	varies
PMS_DMA_APB_I_3_REG	Internal DMA status register.	0x0098	RO
PMS_DMA_RX_I_2_REG	RX Copy DMA permission control register 2.	0x00A4	varies
PMS_DMA_RX_I_3_REG	RX Copy DMA status register.	0x00A8	RO
PMS_DMA_TX_I_2_REG	TX Copy DMA permission control register 2.	0x00B4	varies
PMS_DMA_TX_I_3_REG	TX Copy DMA status register.	0x00B8	RO
PMS_APB_PERIPHERAL_INTR_REG	PeriBus2 permission control register.	0x00F4	varies
PMS_APB_PERIPHERAL_STATUS_REG	PeriBus2 peripheral access status register.	0x00F8	RO
PMS_CPU_PERIPHERAL_INTR_REG	PeriBus1 permission control register.	0x00FC	varies
PMS_CPU_PERIPHERAL_STATUS_REG	PeriBus1 peripheral access status register.	0x0100	RO
<b>Version Control Register</b>			
PMS_DATE	Version control register.	0x0FFC	R/W

## 24.6 Registers

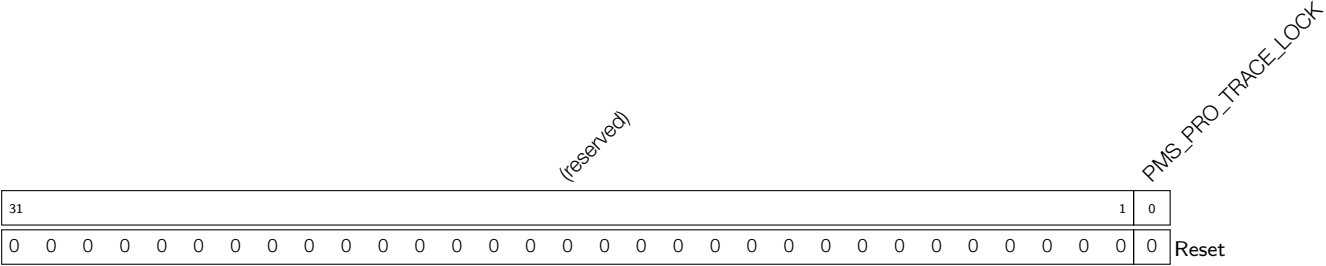
**Register 24.1: PMS\_PRO\_IRAM0\_0\_REG (0x0010)**

(reserved)																															PMS_PRO_IRAM0_LOCK																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**PMS\_PRO\_IRAM0\_LOCK** Lock register. Setting to 1 locks IBUS permission control registers. (R/W)

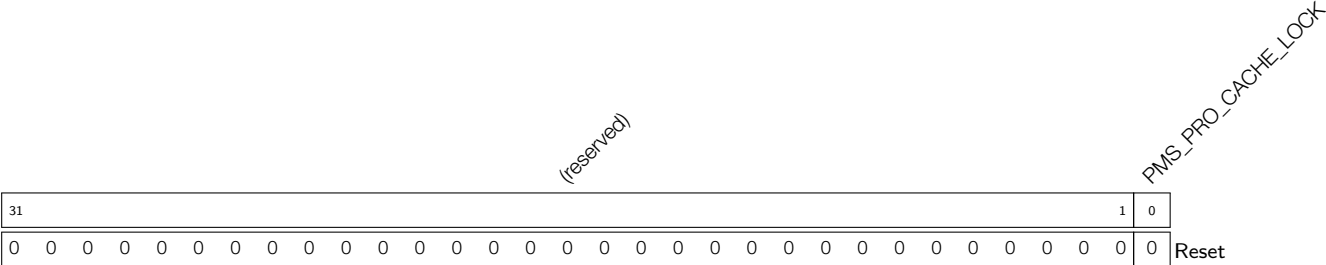


Register 24.5: PMS\_PRO\_TRACE\_0\_REG (0x0070)



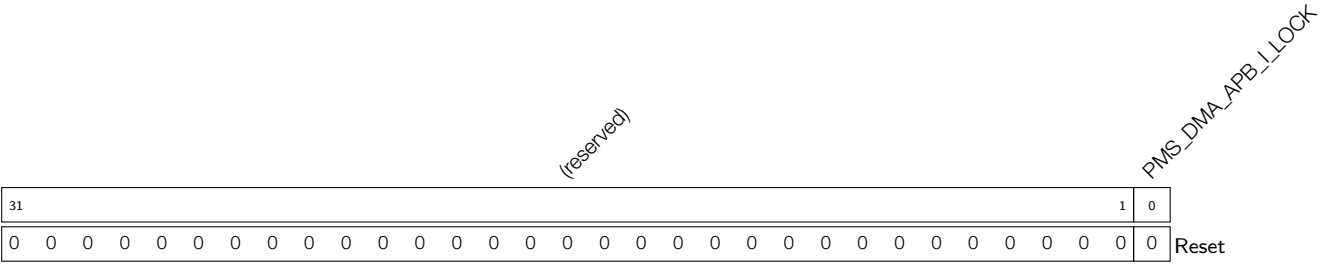
**PMS\_PRO\_TRACE\_LOCK** Lock register. Setting to 1 locks trace function permission control registers. (R/W)

Register 24.6: PMS\_PRO\_CACHE\_0\_REG (0x0078)



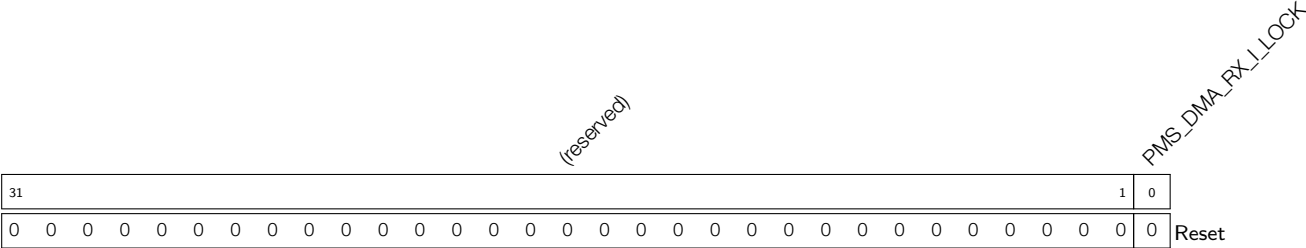
**PMS\_PRO\_CACHE\_LOCK** Lock register. Setting to 1 locks cache permission control registers. (R/W)

Register 24.7: PMS\_DMA\_APB\_I\_0\_REG (0x008C)



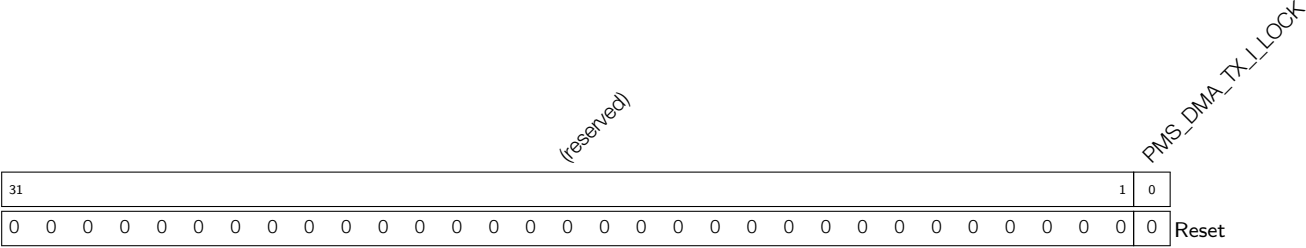
**PMS\_DMA\_APB\_I\_LOCK** Lock register. Setting to 1 locks internal DMA permission control registers. (R/W)

Register 24.8: PMS\_DMA\_RX\_I\_0\_REG (0x009C)



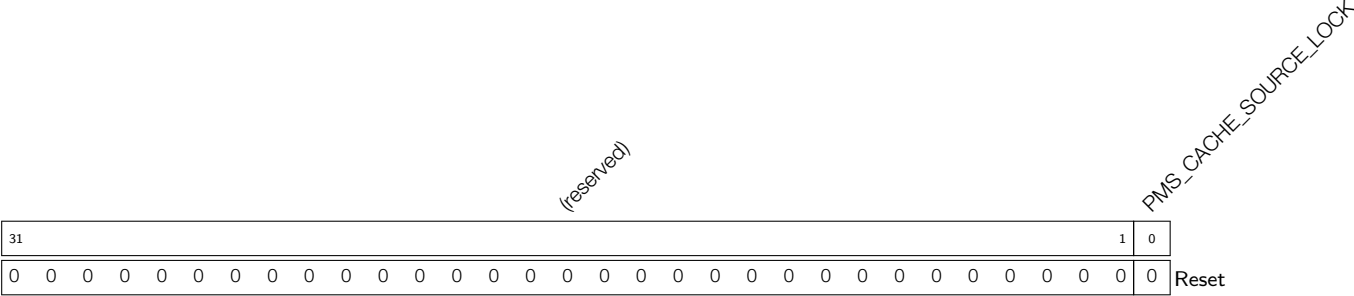
**PMS\_DMA\_RX\_I\_LOCK** Lock register. Setting to 1 locks RX Copy DMA permission control registers. (R/W)

Register 24.9: PMS\_DMA\_TX\_I\_0\_REG (0x00AC)



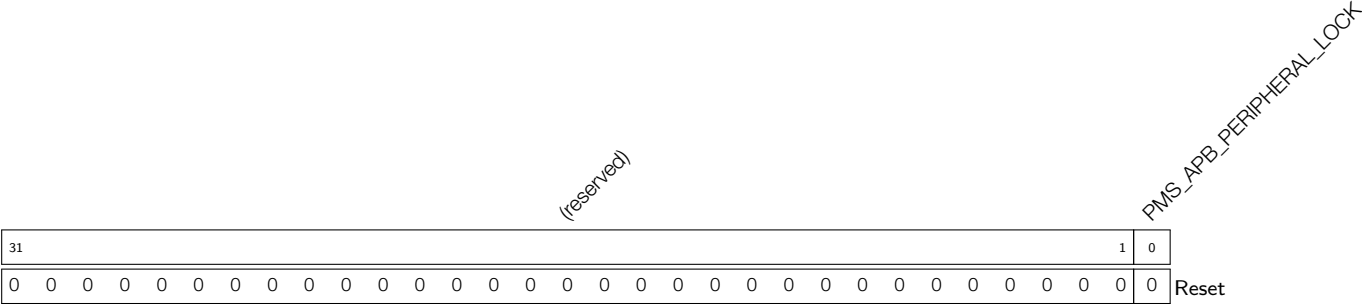
**PMS\_DMA\_TX\_I\_LOCK** Lock register. Setting to 1 locks TX Copy DMA permission control registers. (R/W)

Register 24.10: PMS\_CACHE\_SOURCE\_0\_REG (0x00C4)



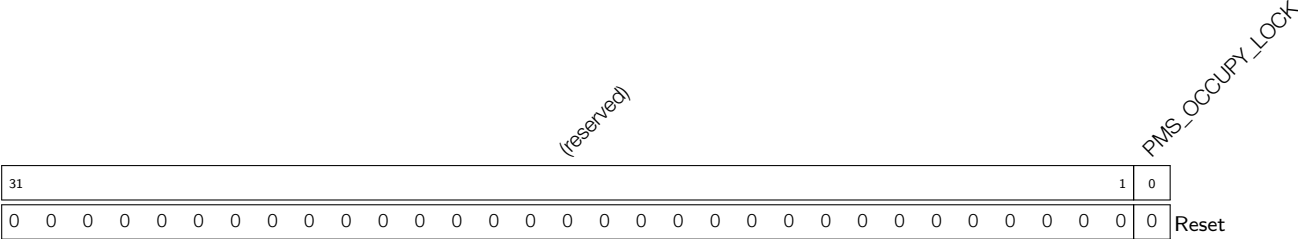
**PMS\_CACHE\_SOURCE\_LOCK** Lock register. Setting to 1 locks cache access permission control registers. (R/W)

Register 24.11: PMS\_APB\_PERIPHERAL\_0\_REG (0x00CC)



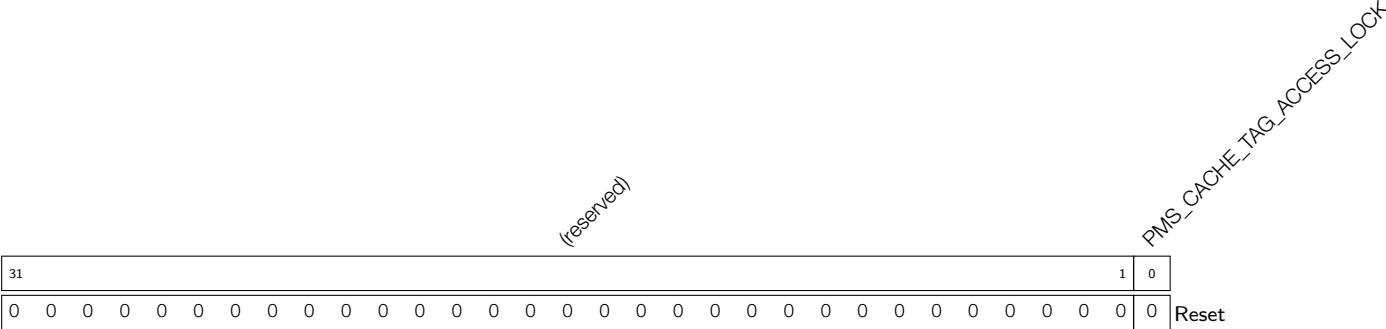
**PMS\_APB\_PERIPHERAL\_LOCK** Lock register. Setting to 1 locks TX Copy DMA permission control registers. (R/W)

Register 24.12: PMS\_OCCUPY\_0\_REG (0x00D4)



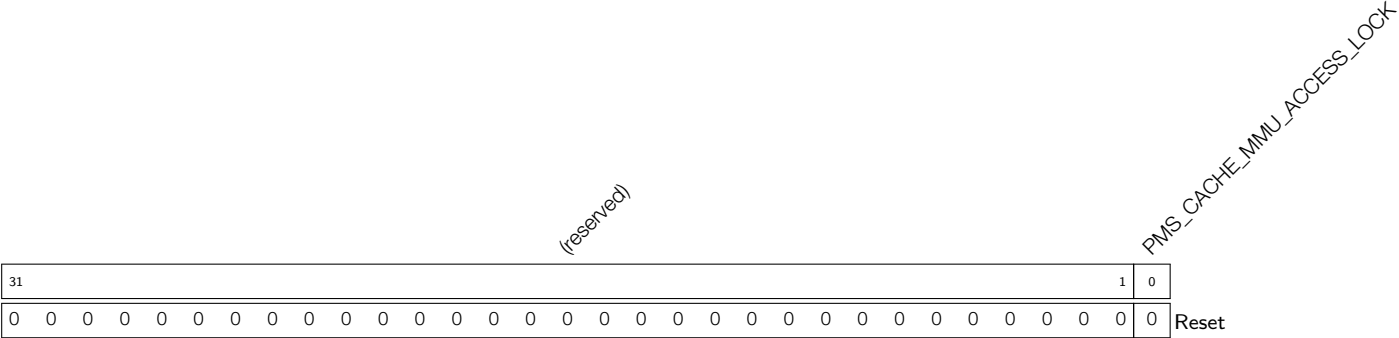
**PMS\_OCCUPY\_LOCK** Lock register. Setting to 1 locks occupy permission control registers. (R/W)

Register 24.13: PMS\_CACHE\_TAG\_ACCESS\_0\_REG (0x00E4)



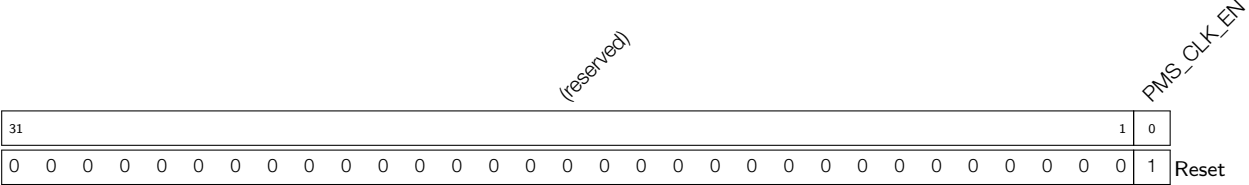
**PMS\_CACHE\_TAG\_ACCESS\_LOCK** Lock register. Setting to 1 locks cache tag permission control registers. (R/W)

Register 24.14: PMS\_CACHE\_MMU\_ACCESS\_0\_REG (0x00EC)



**PMS\_CACHE\_MMU\_ACCESS\_LOCK** Lock register. Setting to 1 locks cache MMU permission control registers. (R/W)

Register 24.15: PMS\_CLOCK\_GATE\_REG\_REG (0x0104)



**PMS\_CLK\_EN** Enable the clock of permission control module when set to 1. (R/W)

Register 24.16: PMS\_PRO\_IRAM0\_1\_REG (0x0014)

(reserved)																								PMS_PRO_IRAM0_SRAM_3_W PMS_PRO_IRAM0_SRAM_3_R PMS_PRO_IRAM0_SRAM_3_F PMS_PRO_IRAM0_SRAM_2_W PMS_PRO_IRAM0_SRAM_2_R PMS_PRO_IRAM0_SRAM_2_F PMS_PRO_IRAM0_SRAM_1_W PMS_PRO_IRAM0_SRAM_1_R PMS_PRO_IRAM0_SRAM_1_F PMS_PRO_IRAM0_SRAM_0_W PMS_PRO_IRAM0_SRAM_0_R PMS_PRO_IRAM0_SRAM_0_F											
31												12												11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	Reset						

**PMS\_PRO\_IRAM0\_SRAM\_0\_F** Setting to 1 grants IBUS permission to fetch SRAM Block 0. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_0\_R** Setting to 1 grants IBUS permission to read SRAM Block 0. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_0\_W** Setting to 1 grants IBUS permission to write SRAM Block 0. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_1\_F** Setting to 1 grants IBUS permission to fetch SRAM Block 1. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_1\_R** Setting to 1 grants IBUS permission to read SRAM Block 1. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_1\_W** Setting to 1 grants IBUS permission to write SRAM Block 1. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_2\_F** Setting to 1 grants IBUS permission to fetch SRAM Block 2. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_2\_R** Setting to 1 grants IBUS permission to read SRAM Block 2. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_2\_W** Setting to 1 grants IBUS permission to write SRAM Block 2. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_3\_F** Setting to 1 grants IBUS permission to fetch SRAM Block 3. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_3\_R** Setting to 1 grants IBUS permission to read SRAM Block 3. (R/W)

**PMS\_PRO\_IRAM0\_SRAM\_3\_W** Setting to 1 grants IBUS permission to write SRAM Block 3. (R/W)

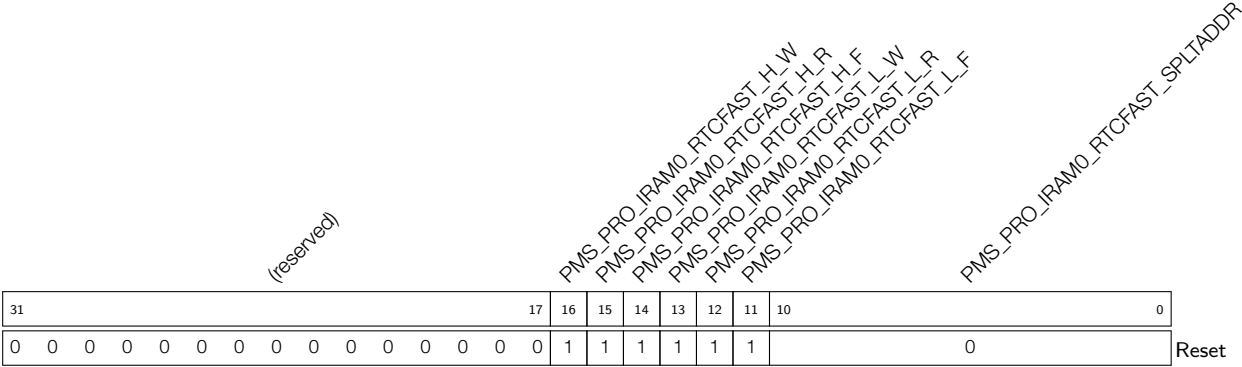
[Submit Documentation Feedback](#)

ESP32-S2 TRM (Preliminary V0.5)

**PMS\_PRO\_IRAM0\_SRAM\_4\_H\_W** Setting to 1 grants IBUS permission to write SRAM Block 4-21 high address region. (R/W)



Register 24.18: PMS\_PRO\_IRAM0\_3\_REG (0x001C)



- PMS\_PRO\_IRAM0\_RTCFAST\_SPLTADDR** Configure the split address of RTC FAST for IBUS access. (R/W)
- PMS\_PRO\_IRAM0\_RTCFAST\_L\_F** Setting to 1 grants IBUS permission to fetch RTC FAST low address region. (R/W)
- PMS\_PRO\_IRAM0\_RTCFAST\_L\_R** Setting to 1 grants IBUS permission to read RTC FAST low address region. (R/W)
- PMS\_PRO\_IRAM0\_RTCFAST\_L\_W** Setting to 1 grants IBUS permission to write RTC FAST low address region. (R/W)
- PMS\_PRO\_IRAM0\_RTCFAST\_H\_F** Setting to 1 grants IBUS permission to fetch RTC FAST high address region. (R/W)
- PMS\_PRO\_IRAM0\_RTCFAST\_H\_R** Setting to 1 grants IBUS permission to read RTC FAST high address region. (R/W)
- PMS\_PRO\_IRAM0\_RTCFAST\_H\_W** Setting to 1 grants IBUS permission to write RTC FAST high address region. (R/W)

Register 24.19: PMS\_PRO\_DRAM0\_1\_REG (0x002C)

(reserved)																PMS_PRO_DRAM0_SRAM_4_H_W																PMS_PRO_DRAM0_SRAM_4_H_R																PMS_PRO_DRAM0_SRAM_4_L_W																PMS_PRO_DRAM0_SRAM_4_L_R																PMS_PRO_DRAM0_SRAM_4_SPLTADDR																PMS_PRO_DRAM0_SRAM_3_W																PMS_PRO_DRAM0_SRAM_3_R																PMS_PRO_DRAM0_SRAM_2_W																PMS_PRO_DRAM0_SRAM_2_R																PMS_PRO_DRAM0_SRAM_1_W																PMS_PRO_DRAM0_SRAM_1_R															
31				29	28	27	26	25	24											8	7	6	5	4	3	2	1	0																																																																																																																																																																			
0	0	0	0	1	1	1	1	0											1	1	1	1	1	1	1	1	1	1	Reset																																																																																																																																																																		

Reset

**PMS\_PRO\_DRAM0\_SRAM\_0\_R** Setting to 1 grants DBUS0 permission to read SRAM Block 0.  
(R/W)

**PMS\_PRO\_DRAM0\_SRAM\_0\_W** Setting to 1 grants DBUS0 permission to write SRAM Block 0.  
(R/W)

**PMS\_PRO\_DRAM0\_SRAM\_1\_R** Setting to 1 grants DBUS0 permission to read SRAM Block 1.  
(R/W)

**PMS\_PRO\_DRAM0\_SRAM\_1\_W** Setting to 1 grants DBUS0 permission to write SRAM Block 1.  
(R/W)

**PMS\_PRO\_DRAM0\_SRAM\_2\_R** Setting to 1 grants DBUS0 permission to read SRAM Block 2.  
(R/W)

**PMS\_PRO\_DRAM0\_SRAM\_2\_W** Setting to 1 grants DBUS0 permission to write SRAM Block 2.  
(R/W)

**PMS\_PRO\_DRAM0\_SRAM\_3\_R** Setting to 1 grants DBUS0 permission to read SRAM Block 3.  
(R/W)

**PMS\_PRO\_DRAM0\_SRAM\_3\_W** Setting to 1 grants DBUS0 permission to write SRAM Block 3.  
(R/W)

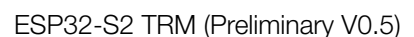
**PMS\_PRO\_DRAM0\_SRAM\_4\_SPLTADDR** Configure the split address of SRAM Block 4-21 for DBUS0 access. (R/W)

**PMS\_PRO\_DRAM0\_SRAM\_4\_L\_R** Setting to 1 grants DBUS0 permission to read SRAM Block 4-21 low address region. (R/W)

**PMS\_PRO\_DRAM0\_SRAM\_4\_L\_W** Setting to 1 grants DBUS0 permission to write SRAM Block 4-21 low address region. (R/W)

**PMS\_PRO\_DRAM0\_SRAM\_4\_H\_R** Setting to 1 grants DBUS0 permission to read SRAM Block 4-21 high address region. (R/W)

**PMS\_PRO\_DRAM0\_SRAM\_4\_H\_W** Setting to 1 grants DBUS0 permission to write SRAM Block 4-21 high address region. (R/W)

[Submit Documentation Feedback](#)

**PMS\_PRO\_DRAM0\_RTCFAST\_H\_W** Setting to 1 grants DBUS0 permission to write RTC FAST high address region. (R/W)

## Register 24.21: PMS\_PRO\_DPORT\_1\_REG (0x0040)

(reserved)																PMS_PRO_DPORT_RESERVE_FIFO_VALID				PMS_PRO_DPORT_RTCSLOW_H_W				PMS_PRO_DPORT_RTCSLOW_H_R				PMS_PRO_DPORT_RTCSLOW_L_W				PMS_PRO_DPORT_RTCSLOW_L_R				PMS_PRO_DPORT_RTCSLOW_SPLTADDR				PMS_PRO_DPORT_APB_PERIPHERAL															
31																20				19				16				15				14				13				12				11				1				0			
0																0				0				1				1				1				1				0				0				Reset							

**PMS\_PRO\_DPORT\_APB\_PERIPHERAL\_FORBID** Setting to 1 denies PeriBus1 bus's access to APB peripheral. (R/W)

**PMS\_PRO\_DPORT\_RTCSLOW\_SPLTADDR** Configure the split address of RTC FAST for PeriBus1 access. (R/W)

**PMS\_PRO\_DPORT\_RTCSLOW\_L\_R** Setting to 1 grants PeriBus1 permission to read RTC FAST low address region. (R/W)

**PMS\_PRO\_DPORT\_RTCSLOW\_L\_W** Setting to 1 grants PeriBus1 permission to write RTC FAST low address region. (R/W)

**PMS\_PRO\_DPORT\_RTCSLOW\_H\_R** Setting to 1 grants PeriBus1 permission to read RTC FAST high address region. (R/W)

**PMS\_PRO\_DPORT\_RTCSLOW\_H\_W** Setting to 1 grants PeriBus1 permission to write RTC FAST high address region. (R/W)

**PMS\_PRO\_DPORT\_RESERVE\_FIFO\_VALID** Configure whether to enable read protection for user-configured FIFO address. (R/W)

## Register 24.22: PMS\_PRO\_DPORT\_2\_REG (0x0044)

(reserved)														PMS_PRO_DPORT_RESERVE_FIFO_0																		
31													18	17															0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x000																	Reset

**PMS\_PRO\_DPORT\_RESERVE\_FIFO\_0** Configure read-protection address 0. (R/W)

Register 24.23: PMS\_PRO\_DPORT\_3\_REG (0x0048)

(reserved)														PMS_PRO_DPORT_RESERVE_FIFO_1																		
31														18	17																0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x000																	Reset

**PMS\_PRO\_DPORT\_RESERVE\_FIFO\_1** Configure read-protection address 1. (R/W)

Register 24.24: PMS\_PRO\_DPORT\_4\_REG (0x004C)

(reserved)														PMS_PRO_DPORT_RESERVE_FIFO_2																		
31														18	17																0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x000																	Reset

**PMS\_PRO\_DPORT\_RESERVE\_FIFO\_2** Configure read-protection address 2. (R/W)

Register 24.25: PMS\_PRO\_DPORT\_5\_REG (0x0050)

(reserved)														PMS_PRO_DPORT_RESERVE_FIFO_3																				
31															18	17																		0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x000																	Reset	

**PMS\_PRO\_DPORT\_RESERVE\_FIFO\_3** Configure read-protection address 3. (R/W)

[Submit Documentation Feedback](#)

ESP32-S2 TRM (Preliminary V0.5)

**PMS\_PRO\_AHB\_RTCSLOW\_0\_H\_W** Setting to 1 grants PeriBus2 permission to write RTCSlow\_0 high address region. (R/W)

Register 24.27: PMS\_PRO\_AHB\_2\_REG (0x0064)

(reserved)																PMS_PRO_AHB_RTCSLOW_1_H_W						PMS_PRO_AHB_RTCSLOW_1_H_R						PMS_PRO_AHB_RTCSLOW_1_H_F						PMS_PRO_AHB_RTCSLOW_1_L_W						PMS_PRO_AHB_RTCSLOW_1_L_R						PMS_PRO_AHB_RTCSLOW_1_L_F						PMS_PRO_AHB_RTCSLOW_1_SPLTADDR					
31																17	16	15	14	13	12	11	10																0																		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0																																		
																										Reset																															

**PMS\_PRO\_AHB\_RTCSLOW\_1\_SPLTADDR** Configure the split address of RTCSlow\_1 for PeriBus2 access. (R/W)

**PMS\_PRO\_AHB\_RTCSLOW\_1\_L\_F** Setting to 1 grants PeriBus2 permission to fetch RTCSlow\_1 low address region. (R/W)

**PMS\_PRO\_AHB\_RTCSLOW\_1\_L\_R** Setting to 1 grants PeriBus2 permission to read RTCSlow\_1 low address region. (R/W)

**PMS\_PRO\_AHB\_RTCSLOW\_1\_L\_W** Setting to 1 grants PeriBus2 permission to write RTCSlow\_1 low address region. (R/W)

**PMS\_PRO\_AHB\_RTCSLOW\_1\_H\_F** Setting to 1 grants PeriBus2 permission to fetch RTCSlow\_1 high address region. (R/W)

**PMS\_PRO\_AHB\_RTCSLOW\_1\_H\_R** Setting to 1 grants PeriBus2 permission to read RTCSlow\_1 high address region. (R/W)

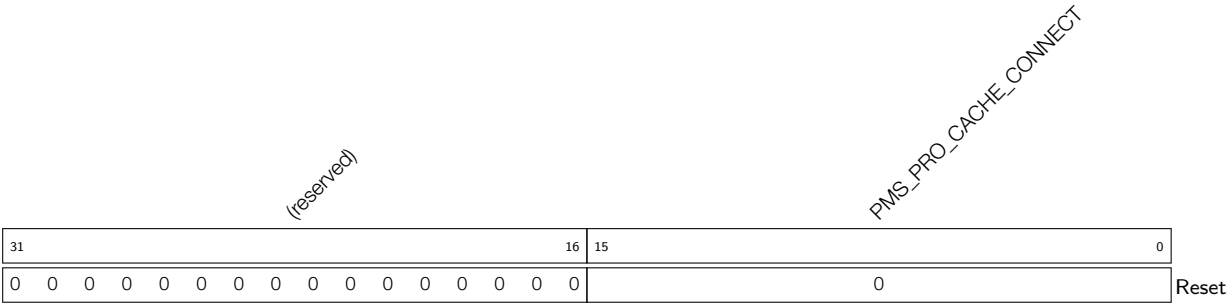
**PMS\_PRO\_AHB\_RTCSLOW\_1\_H\_W** Setting to 1 grants PeriBus2 permission to write RTCSlow\_1 high address region. (R/W)

Register 24.28: PMS\_PRO\_TRACE\_1\_REG (0x0074)

(reserved)																													PMS_PRO_TRACE_DISABLE	
31																													1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**PMS\_PRO\_TRACE\_DISABLE** Setting to 1 disables the trace memory function. (R/W)

Register 24.29: PMS\_PRO\_CACHE\_1\_REG (0x007C)



**PMS\_PRO\_CACHE\_CONNECT** Configure which SRAM Block will be occupied by Icache or Dcache.  
(R/W)



Register 24.30: PMS\_DMA\_APB\_I\_1\_REG (0x0090)

(reserved)								PMS_DMA_APB_I_SRAM_4_H_W								PMS_DMA_APB_I_SRAM_4_H_R								PMS_DMA_APB_I_SRAM_4_L_W								PMS_DMA_APB_I_SRAM_4_L_R																																							
								PMS_DMA_APB_I_SRAM_3_W								PMS_DMA_APB_I_SRAM_3_R								PMS_DMA_APB_I_SRAM_2_W								PMS_DMA_APB_I_SRAM_2_R								PMS_DMA_APB_I_SRAM_1_W								PMS_DMA_APB_I_SRAM_1_R								PMS_DMA_APB_I_SRAM_0_W								PMS_DMA_APB_I_SRAM_0_R							
31	29	28	27	26	25	24												8	7	6	5	4	3	2	1	0																																													
0	0	0	1	1	1	1	0											1	1	1	1	1	1	1	1	1	1	Reset																																											

**PMS\_DMA\_APB\_I\_SRAM\_0\_R** Setting to 1 grants internal DMA permission to read SRAM Block 0.  
(R/W)

**PMS\_DMA\_APB\_I\_SRAM\_0\_W** Setting to 1 grants internal DMA permission to write SRAM Block 0. (R/W)

**PMS\_DMA\_APB\_I\_SRAM\_1\_R** Setting to 1 grants internal DMA permission to read SRAM Block 1.  
(R/W)

**PMS\_DMA\_APB\_I\_SRAM\_1\_W** Setting to 1 grants internal DMA permission to write SRAM Block 1. (R/W)

**PMS\_DMA\_APB\_I\_SRAM\_2\_R** Setting to 1 grants internal DMA permission to read SRAM Block 2.  
(R/W)

**PMS\_DMA\_APB\_I\_SRAM\_2\_W** Setting to 1 grants internal DMA permission to write SRAM Block 2. (R/W)

**PMS\_DMA\_APB\_I\_SRAM\_3\_R** Setting to 1 grants internal DMA permission to read SRAM Block 3.  
(R/W)

**PMS\_DMA\_APB\_I\_SRAM\_3\_W** Setting to 1 grants internal DMA permission to write SRAM Block 3. (R/W)

**PMS\_DMA\_APB\_I\_SRAM\_4\_SPLTADDR** Configure the split address of SRAM Block 4-21 for internal DMA access. (R/W)

**PMS\_DMA\_APB\_I\_SRAM\_4\_L\_R** Setting to 1 grants internal DMA permission to read SRAM Block 4-21 low address region. (R/W)

**PMS\_DMA\_APB\_I\_SRAM\_4\_L\_W** Setting to 1 grants internal DMA permission to write SRAM Block 4-21 low address region. (R/W)

**PMS\_DMA\_APB\_I\_SRAM\_4\_H\_R** Setting to 1 grants internal DMA permission to read SRAM Block 4-21 high address region. (R/W)

**PMS\_DMA\_APB\_I\_SRAM\_4\_H\_W** Setting to 1 grants internal DMA permission to write SRAM Block 4-21 high address region. (R/W)

**Register 24.31: PMS\_DMA\_RX\_I\_1\_REG (0x00A0)**

<div>(reserved)</div> <div>PMS_DMA_RX_I_SRAM_4_H_W</div> <div>PMS_DMA_RX_I_SRAM_4_H_R</div> <div>PMS_DMA_RX_I_SRAM_4_L_W</div> <div>PMS_DMA_RX_I_SRAM_4_L_R</div>																	<div>PMS_DMA_RX_I_SRAM_4_SPLTADDR</div>																	<div>PMS_DMA_RX_I_SRAM_3_W</div> <div>PMS_DMA_RX_I_SRAM_3_R</div> <div>PMS_DMA_RX_I_SRAM_2_W</div> <div>PMS_DMA_RX_I_SRAM_2_R</div> <div>PMS_DMA_RX_I_SRAM_1_W</div> <div>PMS_DMA_RX_I_SRAM_1_R</div> <div>PMS_DMA_RX_I_SRAM_0_W</div> <div>PMS_DMA_RX_I_SRAM_0_R</div>																
31	29	28	27	26	25	24											8	7	6	5	4	3	2	1	0																									
0	0	0	1	1	1	1	0										1	1	1	1	1	1	1	1	1	Reset																								

**PMS\_DMA\_RX\_I\_SRAM\_0\_R** Setting to 1 grants RX Copy DMA permission to read SRAM Block 0. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_0\_W** Setting to 1 grants RX Copy DMA permission to write SRAM Block 0. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_1\_R** Setting to 1 grants RX Copy DMA permission to read SRAM Block 1. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_1\_W** Setting to 1 grants RX Copy DMA permission to write SRAM Block 1. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_2\_R** Setting to 1 grants RX Copy DMA permission to read SRAM Block 2. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_2\_W** Setting to 1 grants RX Copy DMA permission to write SRAM Block 2. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_3\_R** Setting to 1 grants RX Copy DMA permission to read SRAM Block 3. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_3\_W** Setting to 1 grants RX Copy DMA permission to write SRAM Block 3. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_4\_SPLTADDR** Configure the split address of SRAM Block 4-21 for RX Copy DMA access. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_4\_L\_R** Setting to 1 grants RX Copy DMA permission to read SRAM Block 4-21 low address region. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_4\_L\_W** Setting to 1 grants RX Copy DMA permission to write SRAM Block 4-21 low address region. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_4\_H\_R** Setting to 1 grants RX Copy DMA permission to read SRAM Block 4-21 high address region. (R/W)

**PMS\_DMA\_RX\_I\_SRAM\_4\_H\_W** Setting to 1 grants RX Copy DMA permission to write SRAM Block 4-21 high address region. (R/W)

Register 24.32: PMS\_DMA\_TX\_I\_1\_REG (0x00B0)

(reserved)								PMS_DMA_TX_I_SRAM_4_H_W								PMS_DMA_TX_I_SRAM_4_H_R								PMS_DMA_TX_I_SRAM_4_L_W								PMS_DMA_TX_I_SRAM_4_L_R																																																							
								PMS_DMA_TX_I_SRAM_4_SPLTADDR																PMS_DMA_TX_I_SRAM_3_W								PMS_DMA_TX_I_SRAM_3_R								PMS_DMA_TX_I_SRAM_2_W								PMS_DMA_TX_I_SRAM_2_R								PMS_DMA_TX_I_SRAM_1_W								PMS_DMA_TX_I_SRAM_1_R								PMS_DMA_TX_I_SRAM_0_W								PMS_DMA_TX_I_SRAM_0_R							
31	29	28	27	26	25	24												8	7	6	5	4	3	2	1	0																																																													
0	0	0	1	1	1	1	0												1	1	1	1	1	1	1	1	1	1	Reset																																																										

**PMS\_DMA\_TX\_I\_SRAM\_0\_R** Setting to 1 grants TX Copy DMA permission to read SRAM Block 0.  
(R/W)

**PMS\_DMA\_TX\_I\_SRAM\_0\_W** Setting to 1 grants TX Copy DMA permission to write SRAM Block 0.  
(R/W)

**PMS\_DMA\_TX\_I\_SRAM\_1\_R** Setting to 1 grants TX Copy DMA permission to read SRAM Block 1.  
(R/W)

**PMS\_DMA\_TX\_I\_SRAM\_1\_W** Setting to 1 grants TX Copy DMA permission to write SRAM Block 1.  
(R/W)

**PMS\_DMA\_TX\_I\_SRAM\_2\_R** Setting to 1 grants TX Copy DMA permission to read SRAM Block 2.  
(R/W)

**PMS\_DMA\_TX\_I\_SRAM\_2\_W** Setting to 1 grants TX Copy DMA permission to write SRAM Block 2.  
(R/W)

**PMS\_DMA\_TX\_I\_SRAM\_3\_R** Setting to 1 grants TX Copy DMA permission to read SRAM Block 3.  
(R/W)

**PMS\_DMA\_TX\_I\_SRAM\_3\_W** Setting to 1 grants TX Copy DMA permission to write SRAM Block 3.  
(R/W)

**PMS\_DMA\_TX\_I\_SRAM\_4\_SPLTADDR** Configure the split address of SRAM Block 4-21 for TX Copy DMA access. (R/W)

**PMS\_DMA\_TX\_I\_SRAM\_4\_L\_R** Setting to 1 grants TX Copy DMA permission to read SRAM Block 4-21 low address region. (R/W)

**PMS\_DMA\_TX\_I\_SRAM\_4\_L\_W** Setting to 1 grants TX Copy DMA permission to write SRAM Block 4-21 low address region. (R/W)

**PMS\_DMA\_TX\_I\_SRAM\_4\_H\_R** Setting to 1 grants TX Copy DMA permission to read SRAM Block 4-21 high address region. (R/W)

**PMS\_DMA\_TX\_I\_SRAM\_4\_H\_W** Setting to 1 grants TX Copy DMA permission to write SRAM Block 4-21 high address region. (R/W)

Register 24.33: PMS\_APB\_PERIPHERAL\_1\_REG (0x00D0)

(reserved)																														PMS_APB_PERIPHERAL_SPLIT_BURST	
31																														1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Reset

**PMS\_APB\_PERIPHERAL\_SPLIT\_BURST** Setting to 1 splits the data phase of the last access and the address phase of following access. (R/W)

Register 24.34: PMS\_OCCUPY\_1\_REG (0x00D8)

(reserved)																												PMS_OCCUPY_CACHE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
31																												4	3	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**PMS\_OCCUPY\_CACHE** Configure whether SRAM Block 0-3 is used as cache memory. (R/W)

Register 24.35: PMS\_OCCUPY\_3\_REG (0x00E0)

(reserved)														PMS_OCCUPY_PRO_TRACE																		
31														18	17																0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																	Reset

**PMS\_OCCUPY\_PRO\_TRACE** Configure one out block of Block 4-21 is used as trace memory. (R/W)

## Register 24.36: PMS\_CACHE\_TAG\_ACCESS\_1\_REG (0x00E8)

(reserved)																										PMS_PRO_D_TAG_WR_ACS PMS_PRO_D_TAG_RD_ACS PMS_PRO_I_TAG_WR_ACS PMS_PRO_I_TAG_RD_ACS					
31																									4	3	2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**PMS\_PRO\_I\_TAG\_RD\_ACS** Setting to 1 permits read access to Icache tag memory. (R/W)

**PMS\_PRO\_I\_TAG\_WR\_ACS** Setting to 1 permits write access to Icache tag memory. (R/W)

**PMS\_PRO\_D\_TAG\_RD\_ACS** Setting to 1 permits read access to Dcache tag memory. (R/W)

**PMS\_PRO\_D\_TAG\_WR\_ACS** Setting to 1 permits write access to Dcache tag memory. (R/W)

## Register 24.37: PMS\_CACHE\_MMU\_ACCESS\_1\_REG (0x00F0)

(reserved)																								PMS_PRO_MMU_WR_ACS PMS_PRO_MMU_RD_ACS				
31																								2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	Reset

**PMS\_PRO\_MMU\_RD\_ACS** Setting to 1 permits read access to MMU memory. (R/W)

**PMS\_PRO\_MMU\_WR\_ACS** Setting to 1 permits write access to MMU memory. (R/W)

## Register 24.38: PMS\_PRO\_IRAM0\_4\_REG (0x0020)

(reserved)																								PMS_PRO_IRAM0_ILG_INTR PMS_PRO_IRAM0_ILG_EN PMS_PRO_IRAM0_ILG_CLR						
31																									3	2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**PMS\_PRO\_IRAM0\_ILG\_CLR** The clear signal for IBUS access interrupt. (R/W)

**PMS\_PRO\_IRAM0\_ILG\_EN** The enable signal for IBUS access interrupt. (R/W)

**PMS\_PRO\_IRAM0\_ILG\_INTR** IBUS access interrupt signal. (RO)

Register 24.39: PMS\_PRO\_IRAM0\_5\_REG (0x0024)

(reserved)										PMS_PRO_IRAM0_ILG_ST																																									
31										22										21																						0									
0 0 0 0 0 0 0 0 0 0										0																						Reset																			

**PMS\_PRO\_IRAM0\_ILG\_ST** Record the illegitimate information of IBUS. [21:2]: store the bits [21:2] of IBUS address; [1]: 1 means data access, 0 means instruction access; [0]: 1 means write operation, 0 means read operation. (RO)

Register 24.40: PMS\_PRO\_DRAM0\_3\_REG (0x0034)

(reserved)																						PMS_PRO_DRAM0_ILG_INTR PMS_PRO_DRAM0_ILG_EN PMS_PRO_DRAM0_ILG_CLR					
31																								3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**PMS\_PRO\_DRAM0\_ILG\_CLR** The clear signal for DBUS0 access interrupt. (R/W)

**PMS\_PRO\_DRAM0\_ILG\_EN** The enable signal for DBUS0 access interrupt. (R/W)

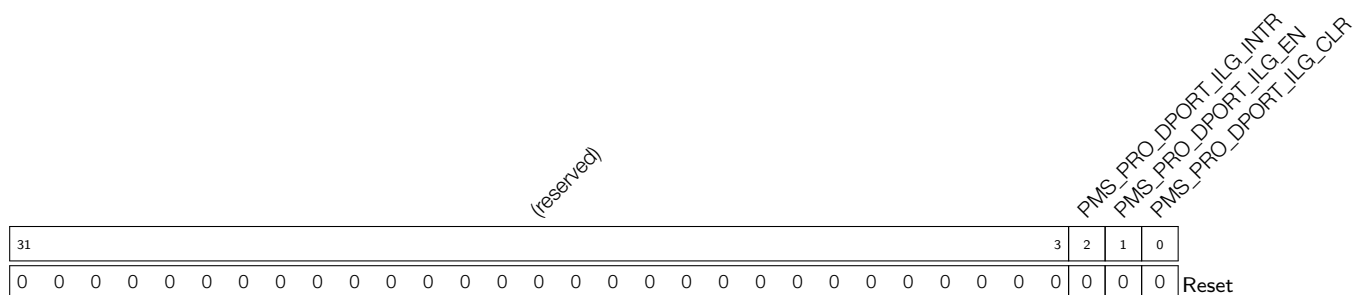
**PMS\_PRO\_DRAM0\_ILG\_INTR** DBUS0 access interrupt signal. (RO)

Register 24.41: PMS\_PRO\_DRAM0\_4\_REG (0x0038)

(reserved)							PMS_PRO_DRAM0_ILG_ST																																				
31							26							25																									0				
0							0							0							0							0							0							0	Reset

**PMS\_PRO\_DRAM0\_ILG\_ST** Record the illegitimate information of DBUS. [25:6]: store the bits [21:2] of DBUS address; [5]: 1 means atomic access, 0 means nonatomic access; [4]: 1 means write operation, 0 means read operation; [3:0]: DBUS0 bus byte enables. (RO)

## Register 24.42: PMS\_PRO\_DPORT\_6\_REG (0x0054)

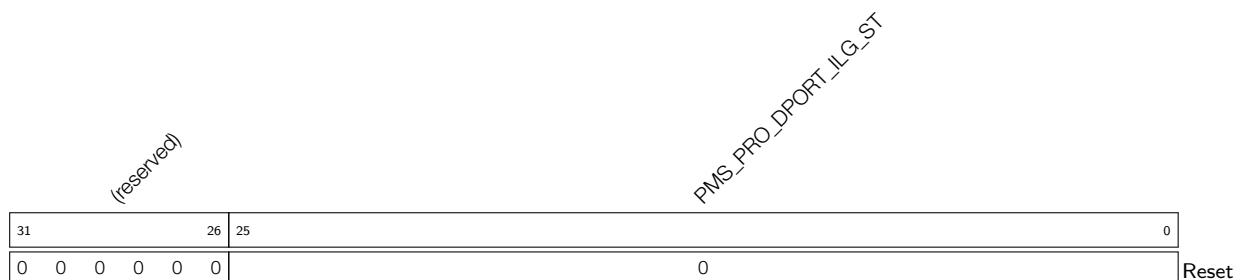


**PMS\_PRO\_DPORT\_ILG\_CLR** The clear signal for PeriBus1 access interrupt. (R/W)

**PMS\_PRO\_DPORT\_ILG\_EN** The enable signal for PeriBus1 access interrupt. (R/W)

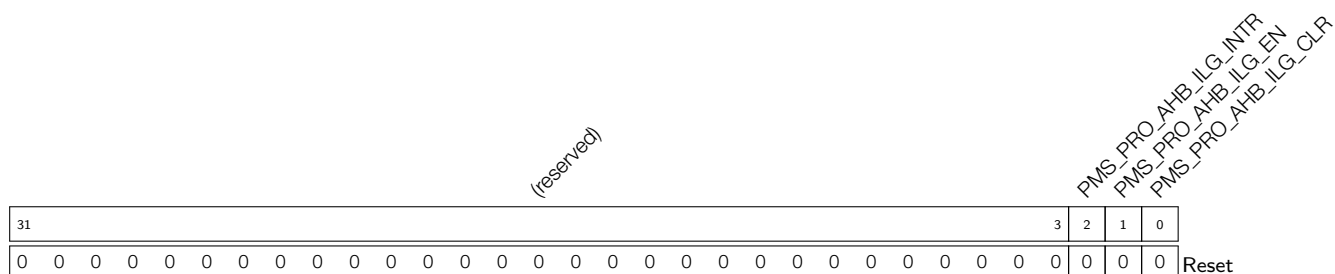
**PMS\_PRO\_DPORT\_ILG\_INTR** PeriBus1 access interrupt signal. (RO)

## Register 24.43: PMS\_PRO\_DPORT\_7\_REG (0x0058)



**PMS\_PRO\_DPORT\_ILG\_ST** Record the illegitimate information of PeriBus1. [25:6]: store the bits [21:2] of PeriBus1 address; [5]: 1 means atomic access, 0 means nonatomic access; [4]: if bits [31:22] of PeriBus1 address are 0xfd, then the bit value is 1, otherwise it is 0; [3:0]: PeriBus 1 byte enables. (RO)

## Register 24.44: PMS\_PRO\_AHB\_3\_REG (0x0068)



**PMS\_PRO\_AHB\_ILG\_CLR** The clear signal for PeriBus2 access interrupt. (R/W)

**PMS\_PRO\_AHB\_ILG\_EN** The enable signal for PeriBus2 access interrupt. (R/W)

**PMS\_PRO\_AHB\_ILG\_INTR** PeriBus2 access interrupt signal. (RO)

[Submit Documentation Feedback](#)



(reserved)



**PMS\_PRO\_CACHE\_ILG\_INTR** Cache access interrupt signal. (RO)

(reserved)



544



**Register 24.48: PMS\_PRO\_CACHE\_4\_REG (0x0088)**

(reserved)																PMS_PRO_CACHE_ILG_ST_D																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																17																16																0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0															

Reset

**PMS\_PRO\_CACHE\_ILG\_ST\_D** Record the illegitimate information of Dcache to access memory.  
 [16]: access enable, active low; [15:4]: store the bits [11:0] of address; [3:0]: Dcache bus byte enables, active low. (RO)

**Register 24.49: PMS\_DMA\_APB\_I\_2\_REG (0x0094)**

(reserved)																								PMS_DMA_APB_I_ILG_INTR PMS_DMA_APB_I_ILG_EN PMS_DMA_APB_I_ILG_CLR			
31																							3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Reset

**PMS\_DMA\_APB\_I\_ILG\_CLR** The clear signal for internal DMA access interrupt. (R/W)

**PMS\_DMA\_APB\_I\_ILG\_EN** The enable signal for internal DMA access interrupt. (R/W)

**PMS\_DMA\_APB\_I\_ILG\_INTR** Internal DMA access interrupt signal. (RO)

**Register 24.50: PMS\_DMA\_APB\_I\_3\_REG (0x0098)**

(reserved)										PMS_DMA_APB_ILG_ST																																
31										23										22																						0
0 0 0 0 0 0 0 0 0 0										0																						Reset										

Reset

**PMS\_DMA\_APB\_I\_ILG\_ST** Record the illegal information of Internal DMA. [22:6]: store the bits [18:2] of address; [5]: if bits [31:19] of address are 0x7ff, then the bit value is 1, otherwise it is 0; [4]: 1 means write operation, 0 means read operation; [3:0]: Internal DMA bus byte enable; (RO)

**Register 24.51: PMS\_DMA\_RX\_I\_2\_REG (0x00A4)**

(reserved)																										PMS_DMA_RX_I_ILG_INTR PMS_DMA_RX_I_ILG_EN PMS_DMA_RX_I_ILG_CLR				
31																										3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**PMS\_DMA\_RX\_I\_ILG\_CLR** The clear signal for RX Copy DMA access interrupt. (R/W)

**PMS\_DMA\_RX\_I\_ILG\_EN** The enable signal for RX Copy DMA access interrupt. (R/W)

**PMS\_DMA\_RX\_I\_ILG\_INTR** RX Copy DMA access interrupt signal. (RO)

**Register 24.52: PMS\_DMA\_RX\_I\_3\_REG (0x00A8)**

(reserved)										PMS_DMA_RX_ILG_ST																																								
31										23																				22																				0
0	0	0	0	0	0	0	0	0	0	0																																						Reset		

**PMS\_DMA\_RX\_I\_ILG\_ST** Record the illegitimate information of RX Copy DMA. [22:6]: store the bits [18:2] of address; [5]: if bits [31:19] of address are 0x7ff, then the bit value is 1, otherwise it is 0; [4]: 1 means write operation, 0 means read operation; [3:0]: RX Copy DMA bus byte enables. (RO)

**Register 24.53: PMS\_DMA\_TX\_I\_2\_REG (0x00B4)**

(reserved)																										PMS_DMA_TX_I_ILG_INTR PMS_DMA_TX_I_ILG_EN PMS_DMA_TX_I_ILG_CLR				
31																										3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**PMS\_DMA\_TX\_I\_ILG\_CLR** The clear signal for TX Copy DMA access interrupt. (R/W)

**PMS\_DMA\_TX\_I\_ILG\_EN** The enable signal for TX Copy DMA access interrupt. (R/W)

**PMS\_DMA\_TX\_I\_ILG\_INTR** TX Copy DMA access interrupt signal. (RO)

Register 24.54: PMS\_DMA\_TX\_I\_3\_REG (0x00B8)

(reserved)										PMS_DMA_TX_I_ILG_ST																																
31										23										22																						0
0 0 0 0 0 0 0 0 0 0										0																						Reset										

**PMS\_DMA\_TX\_I\_ILG\_ST** Record the illegitimate information of TX Copy DMA. [22:6]: store the bits [18:2] of address; [5]: if bits [31:19] of address are 0x7ff, then the bit value is 1, otherwise it is 0; [4]: 1 means write operation, 0 means read operation; [3:0]: TX Copy DMA bus byte enables. (RO)

Register 24.55: PMS\_APB\_PERIPHERAL\_INTR\_REG (0x00F4)

(reserved)																						PMS_APB_PERI_BYTE_ERROR_INTR PMS_APB_PERI_BYTE_ERROR_EN PMS_APB_PERI_BYTE_ERROR_CLR					
31																								3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**PMS\_APB\_PERI\_BYTE\_ERROR\_CLR** The clear signal for APB peripheral interrupt. (R/W)

**PMS\_APB\_PERI\_BYTE\_ERROR\_EN** The enable signal for APB peripheral access interrupt. (R/W)

**PMS\_APB\_PERI\_BYTE\_ERROR\_INTR** APB peripheral access interrupt signal. (RO)

Register 24.56: PMS\_APB\_PERIPHERAL\_STATUS\_REG (0x00F8)

PMS_APB_PERI_BYTE_ERROR_ADDR																											
31																											0
																											Reset

**PMS\_APB\_PERI\_BYTE\_ERROR\_ADDR** Record the illegitimate address of APB peripheral. (RO)

## 548

[Submit Documentation Feedback](#)

## 25. Digital Signature

### 25.1 Overview

Digital signatures provide a way to cryptographically authenticate a message using a private key, to be verified using the corresponding public key. This can be used to validate a device's identity to a server, or to authenticate the integrity of a message has not been tampered with.

ESP32-S2 includes a digital signature (DS) peripheral which produces hardware accelerated RSA digital signatures, without the RSA private key being accessible by software.

### 25.2 Features

- RSA Digital Signatures with key lengths up to 4096 bits
- Private key data is encrypted and only readable by DS peripheral
- SHA-256 digest is used to protect private key data against tampering by an attacker

### 25.3 Functional Description

#### 25.3.1 Overview

The DS peripheral calculates the RSA encryption operation  $Z = X^Y \bmod M$  where  $Z$  is the signature,  $X$  is the input message,  $Y$  and  $M$  are the RSA private key parameters.

Private key parameters are stored in flash or another form of storage, in an encrypted form. They are encrypted using a key which can only be read by the DS peripheral via the HMAC peripheral. The required inputs to generate the key are stored in eFuse and can only be accessed by the HMAC peripheral. This means that only the DS peripheral hardware can decrypt the private key, and the plaintext private key data is never accessed by software.

The input message  $X$  is input directly to the DS peripheral by software, each time a signature is needed. After the operation, the signature  $Z$  is read back by software.

#### 25.3.2 Private Key Operands

Private key operands  $Y$  (private key exponent) and  $M$  (key modulus) are generated by the user. They will have a particular RSA key length (up to 4096 bits). A corresponding public key is also generated and stored separately, it can be used independently to verify DS signatures.

Two additional private key operands are needed —  $\bar{r}$  and  $M'$ . These two operands are derived from  $Y$  and  $M$ , but they are calculated in advance by software.

Operands  $Y$ ,  $M$ ,  $\bar{r}$ , and  $M'$  are encrypted by the user along with an authentication digest and stored as a single ciphertext  $C$ .  $C$  is input to the DS peripheral in this encrypted format, then the hardware decrypts  $C$  and uses the key data to generate the signature. Detailed description of the encryption process to prepare  $C$  is provided in Section [25.3.4](#).

The DS peripheral needs to activate RSA to perform  $Z = X^Y \bmod M$ . For detailed information on the RSA algorithm, please refer to Section [21.3.1 Large Number Modular Exponentiation](#) in Chapter [21 RSA Accelerator](#).

### 25.3.3 Conventions

The following sections of this chapter will use the following symbols and functions:

- $1^s$  A bit string that consists of  $s$  “1” bits.
- $[x]_s$  A bit string of length  $s$  bits. If  $x$  is a number ( $x < 2^s$ ), it is represented in little endian byte order in the bit string.  $x$  may be a variable value such as  $[Y]_{4096}$  or as a hexadecimal constant such as  $[0x0C]_8$ . If necessary, the value  $[x]$  is right-padded with 0s to reach  $s$  bits in length. For example:  $[0x5]_4 = 0101$ ,  $[0x5]_8 = 00000101$ ,  $[0x5]_{16} = 0000010100000000$ ,  $[0x13]_8 = 00010011$ ,  $[0x13]_{16} = 0001001100000000$ .
- $||$  A bit string concatenation operator for joining multiple bit strings into a longer bit string.

### 25.3.4 Software Storage of Private Key Data

To store a private key for use with the DS peripheral, users need to complete the following preparations:

- Generate the RSA private key ( $Y$ ,  $M$ ) and associated operands  $\bar{r}$  and  $M'$ , as described in Section 25.3.2.
- Generate a 256-bit HMAC key ( $[HMAC\_KEY]_{256}$ ) that is stored in eFuse. This HMAC key is read by the HMAC peripheral to derive a key, as  $DS\_KEY = \text{HMAC-SHA256}([HMAC\_KEY]_{256}, 1^{256})$ . This key is used to securely encrypt and decrypt the stored RSA private key data.
- Prepare encrypted private key parameters as ciphertext  $C$ , 1584 bytes in length.

Figure 25-1 below describes the preparations at the software level (the left part) and the DS peripheral operation at the hardware level (the right part).

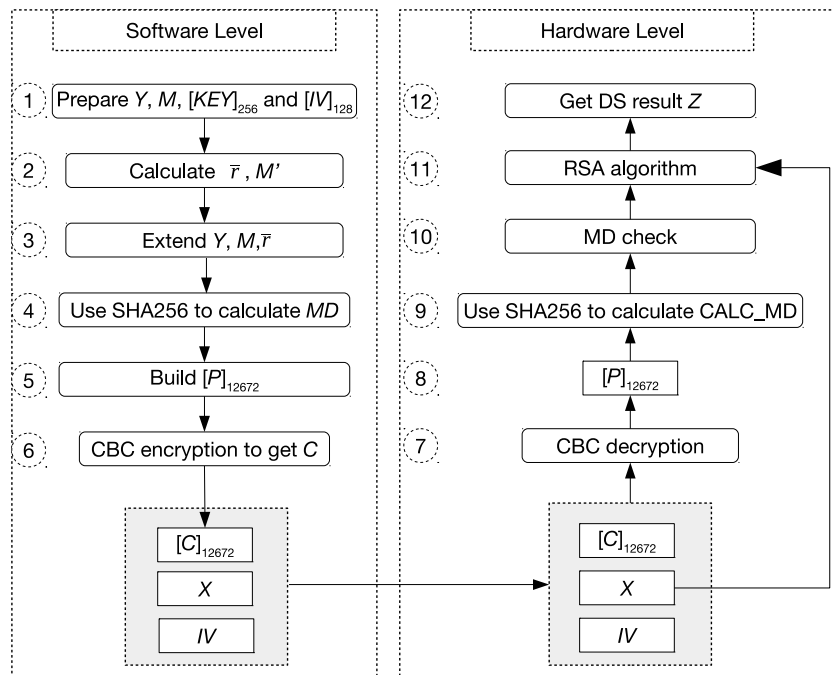


Figure 25-1. Preparations and DS Operation

Users need to follow the steps shown in the left part of Figure 25-1 to calculate  $C$ . Detailed instructions are as follows:

- **Step 1:** Prepare  $Y$  and  $M$  whose lengths should meet the aforementioned requirements. Define  $[L]_{32} = \frac{N}{32}$  (i.e., for RSA 4096,  $[L]_{32} == [0x80]_{32}$ ). Prepare  $[DS\_KEY]_{256}$  and generate a random  $[IV]_{128}$  which

should meet the requirements of the AES-CBC block encryption algorithm. For more information on AES, please refer to Chapter 19 [AES Accelerator](#).

- **Step 2:** Calculate  $\bar{r}$  and  $M'$  based on  $M$ .
- **Step 3:** Extend  $Y$ ,  $M$ , and  $\bar{r}$ , in order to get  $[Y]_{4096}$ ,  $[M]_{4096}$ , and  $[\bar{r}]_{4096}$ , respectively. Since the largest operand length for  $Y$ ,  $M$ , and  $\bar{r}$  is 4096 bits, this step is only required for lengths smaller than 4096 bits.
- **Step 4:** Calculate MD authentication code using the SHA-256 algorithm:  

$$[MD]_{256} = \text{SHA256} ([Y]_{4096} || [M]_{4096} || [\bar{r}]_{4096} || [M']_{32} || [L]_{32} || [IV]_{128})$$
- **Step 5:** Build  $[P]_{12672} = ([Y]_{4096} || [M]_{4096} || [\bar{r}]_{4096} || [MD]_{256} || [M']_{32} || [L]_{32} || [\beta]_{64})$ , where  $[\beta]_{64}$  is a PKCS#7 padding value, i.e., a 64-bit string  $[0x0808080808080808]_{64}$  that is composed of eight bytes (value = 0x08). The purpose of  $[\beta]_{64}$  is to make the bit length of  $P$  a multiple of 128.
- **Step 6:** Calculate  $C = [C]_{12672} = \text{AES-CBC-ENC} ([P]_{12672}, [DS\_KEY]_{256}, [IV]_{128})$ , where  $C$  is the ciphertext that includes RSA operands  $Y$ ,  $M$ ,  $\bar{r}$ ,  $M'$ , and  $L$  as well as the MD authentication code and  $[\beta]_{64}$ .  $DS\_KEY$  is derived from the  $HMAC\_KEY$  stored in eFuse, as described above in Section 25.3.4.

### 25.3.5 DS Operation at the Hardware Level

The hardware operation is triggered each time a Digital Signature needs to be calculated. The inputs are the pre-generated private key ciphertext  $C$ , a unique message  $X$ , and  $IV$ .

The DS operation at the hardware level is a reverse process of preparing  $C$  described in Section 25.3.4. The hardware operation can be divided into the following three stages.

#### 1. Decryption: Step 7 and 8

The decryption process is the reverse of Step 6. The DS peripheral will call AES accelerator to decrypt  $C$  in CBC block mode and get the resulted plaintext. The decryption process can be represented by  $P = \text{AES-CBC-DEC} (C, DS\_KEY, IV)$ , where  $IV$  (i.e.,  $[IV]_{128}$ ) is defined by users.  $[DS\_KEY]_{256}$  is provided by HMAC module, derived from  $HMAC\_KEY$  stored in eFuse.  $[DS\_KEY]_{256}$  is not readable by software.

With  $P$ , the DS peripheral can work out  $[Y]_{4096}$ ,  $[M]_{4096}$ ,  $[\bar{r}]_{4096}$ ,  $[M']_{32}$ ,  $[L]_{32}$ , MD authentication code, and the padding value  $[\beta]_{64}$ . This process is the reverse of Step 5.

#### 2. Check: Step 9 and 10

The DS peripheral will perform two check operations: MD check and padding check. Padding check is not shown in Figure 25-1, as it happens at the same time with MD check.

- MD check: The DS peripheral calls SHA-256 to get the hash value  $[CALC\_MD]_{256}$ . This step is the reverse of Step 4. Then,  $[CALC\_MD]_{256}$  is compared against  $[MD]_{256}$ . Only when the two match, MD check passes.
- Padding check: The DS peripheral checks if  $[\beta]_{64}$  complies with the aforementioned PKCS#7 format. Only when  $[\beta]_{64}$  complies with the format, padding check passes.

If MD check passes, the DS peripheral will perform subsequent operations, otherwise, it will not. If padding check fails, an error bit is set in the query register, but it does not affect the subsequent operations.

#### 3. Calculation: Step 11 and 12

The DS peripheral treats  $X$ ,  $Y$ ,  $M$ , and  $\bar{r}$  as big numbers. With  $M'$ , all operands to perform  $X^Y \bmod M$  are in place. The operand length is defined by  $L$ . The DS peripheral will get the signed result  $Z$  by calling RSA to perform  $Z = X^Y \bmod M$ .

### 25.3.6 DS Operation at the Software Level

The following software steps should be followed each time a Digital Signature needs to be calculated. The inputs are the pre-generated private key ciphertext  $C$ , a unique message  $X$ , and  $IV$ . These software steps trigger the hardware steps described in Section 25.3.5.

1. **Activate the DS peripheral:** Write 1 to [DS\\_SET\\_START\\_REG](#).

2. **Check if  $DS\_KEY$  is ready:** Poll [DS\\_QUERY\\_BUSY\\_REG](#) until it reads 0.

If [DS\\_QUERY\\_BUSY\\_REG](#) does not read 0 after approximately 1 ms, it indicates a problem with HMAC initialization. In such case, software can read register [DS\\_QUERY\\_KEY\\_WRONG\\_REG](#) to get more information.

- If [DS\\_QUERY\\_KEY\\_WRONG\\_REG](#) reads 0, it indicates that HMAC peripheral was not activated.
- If [DS\\_QUERY\\_KEY\\_WRONG\\_REG](#) reads any value from 1 to 15, it indicates that HMAC was activated, but the DS peripheral did not successfully receive the  $DS\_KEY$  value from the HMAC peripheral. This may indicate that the HMAC operation was interrupted due to a software concurrency problem.

3. **Configure register:** Write  $IV$  block to register [DS\\_IV\\_m\\_REG](#) ( $m$ : 0-3). For more information on  $IV$  block, please refer to Chapter 19 [AES Accelerator](#).

4. **Write  $X$  to memory block [DS\\_X\\_MEM](#):** Write  $X_i$  ( $i \in [0, n) \cap \mathbb{N}$ ) to memory block [DS\\_X\\_MEM](#) whose capacity is 128 words. Each word can store one base- $b$  digit. The memory block uses the little endian format for storage, i.e., the least significant digit of the operand is in the lowest address. Words in [DS\\_X\\_MEM](#) block after the configured length of  $X$  ( $N$  bits, as described in Section 25.3.2) are ignored.

5. **Write  $C$  to memory block [DS\\_C\\_MEM](#):** Write  $C_i$  ( $i \in [0, 396) \cap \mathbb{N}$ ) to memory block [DS\\_C\\_MEM](#) whose capacity is 396 words. Each word can store one base- $b$  digit.

6. **Start DS operation:** Write 1 to register [DS\\_SET\\_ME\\_REG](#).

7. **Wait for the operation to be completed:** Poll register [DS\\_QUERY\\_BUSY\\_REG](#) until it reads 0.

8. **Query check result:** Read register [DS\\_QUERY\\_CHECK\\_REG](#) and determine the subsequent operations based on the return value.

- If the value is 0, it indicates that both padding check and MD check pass. Users can continue to get the signed result  $Z$ .
- If the value is 1, it indicates that the padding check passes but MD check fails. The signed result  $Z$  is invalid. The operation would resume directly from Step 10.
- If the value is 2, it indicates that the padding check fails but MD check passes. Users can continue to get the signed result  $Z$ .
- If the value is 3, it indicates that both padding check and MD check fail. The signed result  $Z$  is invalid. The operation would resume directly from Step 10.

9. **Read the signed result:** Read the signed result  $Z_i$  ( $i \in \{0, 1, 2, \dots, n\}$ ) from memory block [DS\\_Z\\_MEM](#). The memory block stores  $Z$  in little-endian byte order.



10. **Exit the operation:** Write 1 to [DS\\_SET\\_FINISH\\_REG](#), then poll [DS\\_QUERY\\_BUSY\\_REG](#) until it reads 0.

After the operation, all the input/output registers and memory blocks are cleared.

## 25.4 Base Address

Users can access the DS peripheral with two base addresses, which can be seen in Table 134. For more information about accessing peripherals from different buses please see Chapter 1: [System and Memory](#).

**Table 134: Digital Signature Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F43D000
PeriBUS2	0x6003D000

## 25.5 Memory Blocks

Both the starting address and ending address in the following table are relative to the DS peripheral base addresses provided in Section 25.4.

Name	Description	Size (byte)	Starting Address	Ending Address	Access
DS_C_MEM	Memory block C	1584	0x0000	0x062F	WO
DS_X_MEM	Memory block X	512	0x0800	0x09FF	WO
DS_Z_MEM	Memory block Z	512	0x0A00	0x0BFF	RO

## 25.6 Register Summary

The addresses in the following table are relative to the DS peripheral base addresses provided in Section 25.4.

Name	Description	Address	Access
<b>Configuration Registers</b>			
<a href="#">DS_IV_0_REG</a>	IV block data	0x0630	WO
<a href="#">DS_IV_1_REG</a>	IV block data	0x0634	WO
<a href="#">DS_IV_2_REG</a>	IV block data	0x0638	WO
<a href="#">DS_IV_3_REG</a>	IV block data	0x063C	WO
<b>Status/Control Registers</b>			
<a href="#">DS_SET_START_REG</a>	Activates the DS peripheral	0x0E00	WO
<a href="#">DS_SET_ME_REG</a>	Starts DS operation	0x0E04	WO
<a href="#">DS_SET_FINISH_REG</a>	Ends DS operation	0x0E08	WO
<a href="#">DS_QUERY_BUSY_REG</a>	Status of the DS	0x0E0C	RO
<a href="#">DS_QUERY_KEY_WRONG_REG</a>	Checks the reason why <i>DS_KEY</i> is not ready	0x0E10	RO
<a href="#">DS_QUERY_CHECK_REG</a>	Queries DS check result	0x0814	RO
<b>Version Register</b>			
<a href="#">DS_DATE_REG</a>	Version control register	0x0820	W/R

## 25.7 Registers

Register 25.1: DS\_IV\_ *m*\_REG (*m*: 0-3) (0x0630+4\**m*)

31	0
0x00000000	
Reset	

DS\_IV\_ *m*\_REG (*m*: 0-3) IV block data. (WO)

Register 25.2: DS\_SET\_START\_REG (0x0E00)

(reserved)																														DS_SET_START		
31																														1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Reset																																

DS\_SET\_START Write 1 to this register to activate the DS peripheral. (WO)

Register 25.3: DS\_SET\_ME\_REG (0x0E04)

(reserved)																														DS_SET_ME		
31																														1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Reset																																

DS\_SET\_ME Write 1 to this register to start DS operation. (WO)

Register 25.4: DS\_SET\_FINISH\_REG (0x0E08)

(reserved)																														DS_SET_FINISH		
31																														1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Reset																																

DS\_SET\_FINISH Write 1 to this register to end DS operation. (WO)

**Register 25.5: DS\_QUERY\_BUSY\_REG (0x0E0C)**

(reserved)																														DS_QUERY_BUSY		
31																														1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**DS\_QUERY\_BUSY** 1: The DS peripheral is busy; 0: The DS peripheral is idle. (RO)

**Register 25.6: DS\_QUERY\_KEY\_WRONG\_REG (0x0E10)**

(reserved)																										DS_QUERY_KEY_WRONG					
31																										4	3	0			
0 0																										0x0				Reset	

**DS\_QUERY\_KEY\_WRONG** 1-15: HMAC was activated, but the DS peripheral did not successfully receive the *DS\_KEY* value from the HMAC peripheral. The biggest value is 15. 0: HMAC is not activated. (RO)

**Register 25.7: DS\_QUERY\_CHECK\_REG (0x0E14)**

(reserved)																										DS_PADDING_BAD DS_MD_ERROR					
31																											2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**DS\_PADDING\_BAD** 1: The padding check fails; 0: The padding check passes. (RO)

**DS\_MD\_ERROR** 1: The MD check fails; 0: The MD check passes. (RO)

**Register 25.8: DS\_DATE\_REG (0x0E20)**

(reserved)			DS_DATE																												
31	30	29																												0	
0	0	0x20190418																											Reset		

**DS\_DATE** Version control register. (R/W)

## 26. HMAC Module

### 26.1 Overview

The Hash-based Message Authentication Code (HMAC) module computes Message Authentication Codes (MACs) through Hash algorithms and keys as described in RFC 2104. The underlying hash algorithm is SHA-256, and the 256-bit HMAC key is stored in an eFuse key block and can be configured as not readable by software.

The HMAC module can be used in two modes - in "upstream" mode the HMAC message is supplied by the user and the calculation result is read back by the user. In "downstream" mode the HMAC module is used as a Key Derivation Function (KDF) for other internal hardwares.

### 26.2 Main Features

- Standard HMAC-SHA-256 algorithm
- The hash result is only accessible by the configurable hardware peripheral (downstream mode)
- Supports identity verification challenge-response algorithms
- Supports Digital Signature peripheral (downstream mode)
- Supports re-enabling of Soft-Disabled JTAG (downstream mode)

### 26.3 Functional Description

#### 26.3.1 Upstream Mode

In Upstream mode, the HMAC message is provided by the user and the result is read back by the user.

This allows the key stored in eFuse (can be configured as not readable by software) to become a shared secret between the user and another party. Any challenge-response protocol that supports HMAC-SHA-256 can be used in this way.

The generalized form of these protocols is as follows:

- A calculates a unique nonce message M
- A sends M to B
- B calculates HMAC (M, KEY) and sends to A
- A also calculates HMAC (M, KEY) internally
- A compares both results, If the same, then the identity of B is verified

To set up the key:

1. A 256-bit HMAC key is randomly generated and is programmed to an eFuse key block with corresponding purpose eFuse set to EFUSE\_KEY\_PURPOSE\_HMAC\_UP. See Chapter 16 for details.
2. Configure the eFuse key block to be read protected, so software cannot read back the value. A copy of this secret key should be kept by any other party that wants to authenticate this device.

To calculate an HMAC:

1. User initializes the HMAC module, and enter upstream mode.

2. User writes the correctly padded message to the peripheral, one block at a time.
3. User reads back the HMAC result from peripheral registers.

See Section [26.3.5](#) for detailed steps of this process.

### 26.3.2 Downstream JTAG Enable Mode

eFuse memory has two parameters to disable JTAG debugging: EFUSE\_HARD\_DIS\_JTAG and EFUSE\_SOFT\_DIS\_JTAG. JTAG will be disabled permanently if the former is programmed to 1, and it will be disabled temporarily if the latter is programmed to 1. See Chapter [16](#) for details.

The HMAC peripheral can be used to re-enable JTAG when EFUSE\_SOFT\_DIS\_JTAG is programmed.

To set up the key:

1. A 256-bit HMAC key is randomly generated and is programmed to an eFuse key block with purpose set to either EFUSE\_KEY\_PURPOSE\_HMAC\_DOWN\_JTAG or EFUSE\_KEY\_PURPOSE\_HMAC\_DOWN\_ALL. In the latter case, the same key can be used for both DS and JTAG re-enable functions.
2. Configure the eFuse key block to be read protected, so software cannot read back the value. User stores the randomly generated HMAC key in the process securely elsewhere.
3. Program the eFuse EFUSE\_SOFT\_DIS\_JTAG to 1.

To re-enable JTAG:

1. User performs HMAC calculation on the 32-byte 0x00 locally using SHA-256 and the known random key, and inputs this pre-calculated value into the registers  
SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_0 ~  
SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_7.
2. User enables the HMAC module, and enters downstream JTAG enable mode.
3. If the HMAC calculated result matches the value supplied in the registers  
SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_0 ~  
SYSTEM\_CANCEL\_EFUSE\_DISABLE\_JTAG\_TEMPORARY\_7, JTAG is re-enabled. Otherwise, JTAG remains disabled.
4. JTAG remains the status in step 3 until the user writes 1 in register [HMAC\\_SET\\_INVALIDATE\\_JTAG\\_REG](#), or restarts the system.

See Section [26.3.5](#) for detailed steps of this process.

### 26.3.3 Downstream Digital Signature Mode

The Digital Signature (DS) module encrypts its parameters using AES-CBC. The HMAC module is used as a Key Derivation Function (KDF) to derive the AES key used for this encryption.

To set up the key:

1. A 256-bit HMAC key is randomly generated and is programmed to an eFuse key block with purpose set to either EFUSE\_KEY\_PURPOSE\_HMAC\_DOWN\_DIGITAL\_SIGNATURE or EFUSE\_KEY\_PURPOSE\_HMAC\_DOWN\_ALL. In the latter case, the same key can be used for both DS and JTAG re-enable functions.
2. Configure the eFuse key block to be read protected, so software cannot read back the value. If necessary a copy of the key can also be stored in a secure location.

Before using the DS module, software needs to enable the calculation task of HMAC module's downstream DS mode. The above calculation result will be used as the key when the DS mode performs the calculation task. Consult the [25](#) chapter for details.

### 26.3.4 HMAC eFuse Configuration

The correct implementation of the HMAC module depends on whether the selected eFuse key block is consistent with the configured HMAC function.

#### Configure HMAC Function

Currently, HMAC module supports three functions: JTAG re-enable in downstream mode, DS Key Derivation in downstream mode, and HMAC calculation in upstream mode. Table [137](#) lists the configuration register value corresponding to each function. The values corresponding to the function in use should be written into the register [HMAC\\_SET\\_PARA\\_PURPOSE\\_REG](#) (see Section [26.3.5](#)).

**Table 137: HMAC Function and Configuration Value**

Functions	Mode Type	Value	Description
JTAG Re-enable	Downstream	6	EFUSE_KEY_PURPOSE_HMAC_DOWN_JTAG
DS Key Derivation	Downstream	7	EFUSE_KEY_PURPOSE_HMAC_DOWN_DIGITAL_SIGNATURE
HMAC Calculation	Upstream	8	EFUSE_KEY_PURPOSE_HMAC_UP
Both JTAG Re-enable and DS KDF	Downstream	5	EFUSE_KEY_PURPOSE_HMAC_DOWN_ALL

#### Select eFuse Key Blocks

The eFuse controller provides six key blocks, KEY0 ~ 5. To select a particular KEY $n$  for HMAC module use at runtime, user writes the number  $n$  into register [HMAC\\_SET\\_PARA\\_KEY\\_REG](#).

Note that the purpose of the key is also programmed into eFuse memory. Only when the configured HMAC purpose matches the purpose defined in KEY $n$ , will the HMAC module execute the configured computation.

For more information see [16](#) chapter.

For example, suppose the user selects KEY3 for the computation, and the value programmed in KEY\_PURPOSE\_3 is 6 (EFUSE\_KEY\_PURPOSE\_HMAC\_DOWN\_JTAG). Based on Table [137](#), KEY3 is the key used for JTAG restart. If the configured value of [HMAC\\_SET\\_PARA\\_PURPOSE\\_REG](#) is also 6, then the HMAC peripheral will allow the JTAG start the computation of JTAG re-enable.

### 26.3.5 HMAC Process (Detailed)

The process to call HMAC in ESP32-S2 is as follows:

1. Enable HMAC module
  - (a) Enable the peripheral clock bits for HMAC and SHA peripherals, and clear the corresponding peripheral reset bits.
  - (b) Write 1 into register [HMAC\\_SET\\_START\\_REG](#).
2. Configure HMAC keys and key functions

- (a) Write  $m$  representing key functions into register [HMAC\\_SET\\_PARA\\_PURPOSE\\_REG](#). Correlation between value  $m$  and key functions is shown in Table 58. Refer to Section 26.3.4.
- (b) Select KEY $n$  of eFuse memory as the key by writing  $n$  into register [HMAC\\_SET\\_PARA\\_KEY\\_REG](#) ( $n$  in the range of 0 to 5). Refer to Section 26.3.4.
- (c) Finish the configuration by writing 1 into register [HMAC\\_SET\\_PARA\\_FINISH\\_REG](#).
- (d) Read register [HMAC\\_QUERY\\_ERROR\\_REG](#). Value of 1 means the selected key block does not match the configured key purpose, and computation will not proceed. Value of 0 means the selected key block matches the configured key purpose, and computation can proceed.
- (e) Setting [HMAC\\_SET\\_PARA\\_PURPOSE\\_REG](#) to values other than 8 means HMAC module will operate in downstream mode, proceed with Step 3. Setting value 8 means HMAC module will operate in upstream mode, proceed with Step 4.

### 3. Downstream Mode

- (a) Poll state register [HMAC\\_QUERY\\_BUSY\\_REG](#). The register value of 0 means HMAC computation in downstream mode is finished.
- (b) In downstream mode, the result is used by JTAG module or DS module in the hardware. Users can write 1 into register [HMAC\\_SET\\_INVALIDATE\\_JTAG\\_REG](#) to clean the result generated by JTAG key; or can write 1 into register [HMAC\\_SET\\_INVALIDATE\\_DS\\_REG](#) to clean the result generated by digital signature key.
- (c) This is the end of the HMAC downstream operation.

### 4. Upstream Mode Transmit message block Block $_n$ ( $n \geq 1$ )

- (a) Poll state register [HMAC\\_QUERY\\_BUSY\\_REG](#). Go to the next step when the value of the register is 0.
- (b) Write 512-bit message block Block $_n$  into register range [HMAC\\_WDATA0~15\\_REG](#). Then write 1 in register [HMAC\\_SET\\_MESSAGE\\_ONE\\_REG](#), and HMAC module will compute this message block.
- (c) Poll state register [HMAC\\_QUERY\\_BUSY\\_REG](#). Go to the next step when the value of the register is 0.
- (d) Subsequent message blocks are different, depending on whether the size of to-be-processed data is a multiple of 512 bits.
  - If the bit length of the message is a multiple of 512, there are three possible options:
    - i. If Block $_{n+1}$  exists, write 1 in register [HMAC\\_SET\\_MESSAGE\\_ING\\_REG](#) to make  $n = n + 1$ , then jump to step 4.(b).
    - ii. If Block $_n$  is the last block of the message, and user wishes to apply SHA padding through hardware, write 1 in register [HMAC\\_SET\\_MESSAGE\\_END\\_REG](#), then jump to step 6.
    - iii. If Block $_n$  is the last block of the padded message, and the user has applied SHA padding in software, write 1 in register [HMAC\\_SET\\_MESSAGE\\_PAD\\_REG](#), and jump to step 5.
  - If the bit length of the message is not a multiple of 512, there are three possible options. Note that in this case the user is required to apply SHA padding to the message, after which the padded message length will be a multiple of 512 bits.
    - i. If Block $_n$  is the only message block,  $n = 1$ , and Block $_1$  has included all padding bits, write 1 in register [HMAC\\_ONE\\_BLOCK\\_REG](#), and then jump to step 6.

- ii. If Block\_*n* is the second last block of the padded message, write 1 in register `HMAC_SET_MESSAGE_PAD_REG`, and jump to step 5.
- iii. If Block\_*n* is neither the last nor the second last message block, write 1 in register `HMAC_SET_MESSAGE_ING_REG` and make  $n = n + 1$ , then jump to step 4.(b).

#### 5. Apply SHA Padding to Message

- (a) Users apply SHA padding to the final message block as described in Section 26.4.1, Write this block in register `HMAC_WDATA0~15_REG`, then write 1 in register `HMAC_SET_MESSAGE_ONE_REG`. HMAC module will compute the message block.
- (b) Jump to step 6.

#### 6. Read hash result in upstream mode

- (a) Poll state register `HMAC_QUERY_BUSY_REG`. Go to the next step when the value of the register is 0.
- (b) Read hash result from register `HMAC_RDATA0~7_REG`.
- (c) Write 1 in register `HMAC_SET_RESULT_FINISH_REG` to finish the computation.

#### Note:

The SHA accelerator can be called directly, or used internally by the DS module and the HMAC module. However, they can not share the hardware resources simultaneously. Therefore, SHA module can not be called by CPU or DS module when HMAC module is in use.

## 26.4 HMAC Algorithm Details

### 26.4.1 Padding Bits

HMAC module uses the SHA-256 hash algorithm. If the input message is not a multiple of 512 bits, the user must apply SHA-256 padding algorithm in software. The SHA-256 padding algorithm is described here, and is the same as “5.1 Padding the Message” of “FIPS PUB 180-4”.

As shown in Figure 26-1, suppose the length of the unpadded message is  $m$  bits. Padding steps are as follows:

1. Append a single 1-bit “1” to the end of unpadded message;
2. Append  $k$  bits of value “0”, where  $k$  is the smallest non-negative number which satisfies  $m + 1 + k \equiv 448 \pmod{512}$ ;
3. Append a 64-bit integer value as a binary block. The content of this block is the length of the the unpadded message as a big-endian binary integer value  $m$ .

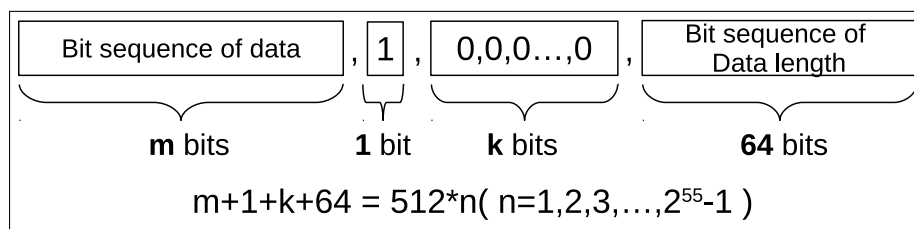


Figure 26-1. HMAC SHA-256 Padding Diagram



In downstream mode, there is no need for users to input any message data or apply padding. In upstream mode, if the total number of bits in the unpadded message is a multiple of 512, users can choose to configure hardware to apply SHA padding. If the total number of bits in the unpadded message is not a multiple of 512, SHA padding must be applied by the user. See Section 26.3.5 for the steps involved.

### 26.4.2 HMAC Algorithm Structure

The Structure of HMAC algorithm as implemented in the HMAC module is shown in Figure 26-2. This is the standard HMAC algorithm as described in RFC 2104.

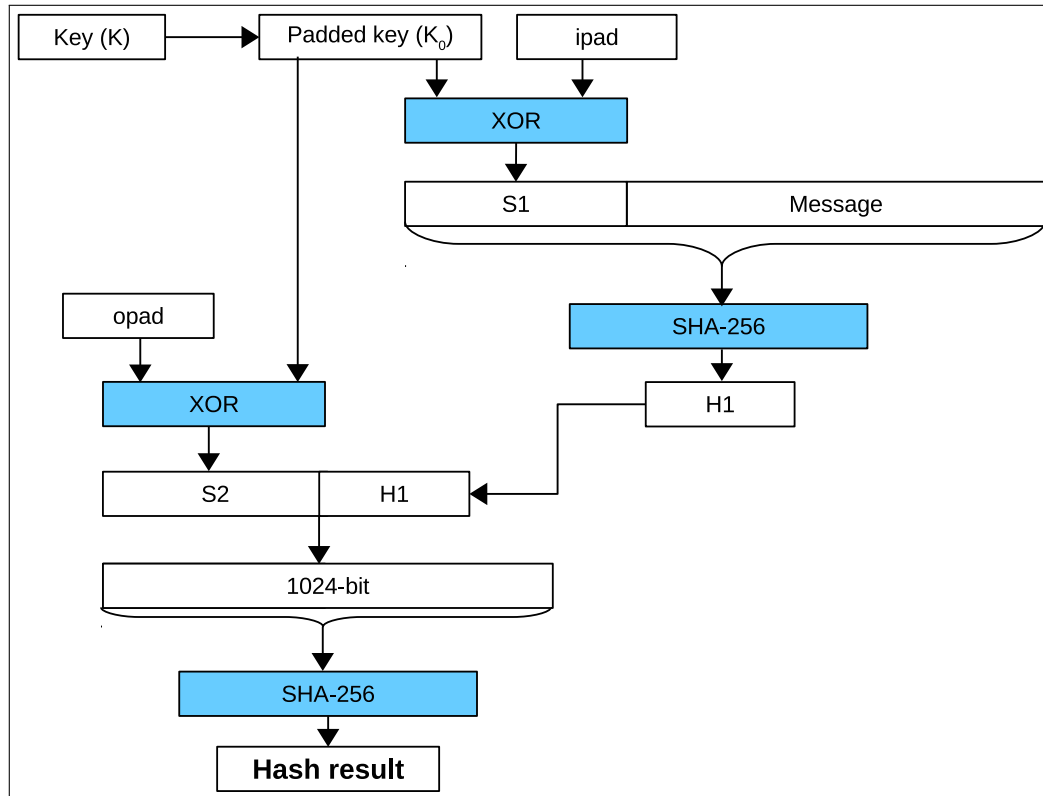


Figure 26-2. HMAC Structure Schematic Diagram

In Figure 26-2,

1. ipad is 512-bit message block composed of sixty-four 0x36 byte.
2. opad is 512-bit message block composed of sixty-four 0x5c byte.

HMAC module appends 256-bit 0 sequence after the bit sequence of 256-bit key  $k$ , and gets 512-bit  $K_0$ . Afterwards, HMAC module XORs  $K_0$  with ipad to get 512-bit  $S_1$ . Then, HMAC module appends the input message which is a multiple of 512 after 512-bit  $S_1$ , and processes SHA-256 algorithm to get 256-bit  $H_1$ .

HMAC module appends the 256-bit SHA-256 hash result to the 512-bit  $S_2$  value, which is calculated by XOR between  $K_0$  and opad, this produces a 768-bit sequence. Then, HMAC module uses SHA padding algorithm described in Section 26.4.1 to pad 768-bit sequence into 1024-bit sequence, and applies SHA-256 algorithm to get the final hash result.

## 26.5 Base Address

Users can access the HMAC module with two base addresses, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter 1: [System and Memory](#).

**Table 138: HMAC Base Address**

Bus to Access Peripheral	Base Address
PeriBUS1	0x3F43E000
PeriBUS2	0x6003E000

## 26.6 Register Summary

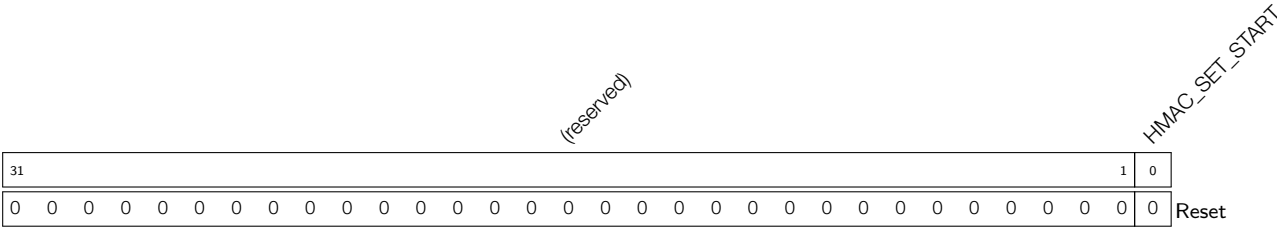
The addresses in the following table are relative to the system registers base addresses provided in Section 26.5.

Name	Description	Address	Access
<b>Control/Status Registers</b>			
<a href="#">HMAC_SET_START_REG</a>	HMAC start control register	0x0040	WO
<a href="#">HMAC_SET_PARA_FINISH_REG</a>	HMAC configuration completion register	0x004C	WO
<a href="#">HMAC_SET_MESSAGE_ONE_REG</a>	HMAC one message control register	0x0050	WO
<a href="#">HMAC_SET_MESSAGE_ING_REG</a>	HMAC message continue register	0x0054	WO
<a href="#">HMAC_SET_MESSAGE_END_REG</a>	HMAC message end register	0x0058	WO
<a href="#">HMAC_SET_RESULT_FINISH_REG</a>	HMAC read result completion register	0x005C	WO
<a href="#">HMAC_SET_INVALIDATE_JTAG_REG</a>	Invalidate JTAG result register	0x0060	WO
<a href="#">HMAC_SET_INVALIDATE_DS_REG</a>	Invalidate digital signature result register	0x0064	WO
<a href="#">HMAC_QUERY_ERROR_REG</a>	The matching result between key and purpose user configured	0x0068	RO
<a href="#">HMAC_QUERY_BUSY_REG</a>	The busy state of HMAC module	0x006C	RO
<b>configuration Registers</b>			
<a href="#">HMAC_SET_PARA_PURPOSE_REG</a>	HMAC parameter configuration register	0x0044	WO
<a href="#">HMAC_SET_PARA_KEY_REG</a>	HMAC key configuration register	0x0048	WO
<b>HMAC Message Block</b>			
<a href="#">HMAC_WR_MESSAGE_0_REG</a>	Message register 0	0x0080	WO
<a href="#">HMAC_WR_MESSAGE_1_REG</a>	Message register 1	0x0084	WO
<a href="#">HMAC_WR_MESSAGE_2_REG</a>	Message register 2	0x0088	WO
<a href="#">HMAC_WR_MESSAGE_3_REG</a>	Message register 3	0x008C	WO
<a href="#">HMAC_WR_MESSAGE_4_REG</a>	Message register 4	0x0090	WO
<a href="#">HMAC_WR_MESSAGE_5_REG</a>	Message register 5	0x0094	WO
<a href="#">HMAC_WR_MESSAGE_6_REG</a>	Message register 6	0x0098	WO
<a href="#">HMAC_WR_MESSAGE_7_REG</a>	Message register 7	0x009C	WO
<a href="#">HMAC_WR_MESSAGE_8_REG</a>	Message register 8	0x00A0	WO
<a href="#">HMAC_WR_MESSAGE_9_REG</a>	Message register 9	0x00A4	WO
<a href="#">HMAC_WR_MESSAGE_10_REG</a>	Message register 10	0x00A8	WO
<a href="#">HMAC_WR_MESSAGE_11_REG</a>	Message register 11	0x00AC	WO
<a href="#">HMAC_WR_MESSAGE_12_REG</a>	Message register 12	0x00B0	WO

Name	Description	Address	Access
<a href="#">HMAC_WR_MESSAGE_13_REG</a>	Message register 13	0x00B4	WO
<a href="#">HMAC_WR_MESSAGE_14_REG</a>	Message register 14	0x00B8	WO
<a href="#">HMAC_WR_MESSAGE_15_REG</a>	Message register 15	0x00BC	WO
<b>HMAC Upstream Result</b>			
<a href="#">HMAC_RD_RESULT_0_REG</a>	Hash result register 0	0x00C0	RO
<a href="#">HMAC_RD_RESULT_1_REG</a>	Hash result register 1	0x00C4	RO
<a href="#">HMAC_RD_RESULT_2_REG</a>	Hash result register 2	0x00C8	RO
<a href="#">HMAC_RD_RESULT_3_REG</a>	Hash result register 3	0x00CC	RO
<a href="#">HMAC_RD_RESULT_4_REG</a>	Hash result register 4	0x00D0	RO
<a href="#">HMAC_RD_RESULT_5_REG</a>	Hash result register 5	0x00D4	RO
<a href="#">HMAC_RD_RESULT_6_REG</a>	Hash result register 6	0x00D8	RO
<a href="#">HMAC_RD_RESULT_7_REG</a>	Hash result register 7	0x00DC	RO
<b>Control/Status registers</b>			
<a href="#">HMAC_SET_MESSAGE_PAD_REG</a>	Software padding register	0x00F0	WO
<a href="#">HMAC_ONE_BLOCK_REG</a>	One block message register	0x00F4	WO
<b>Version Register</b>			
<a href="#">HMAC_DATE_REG</a>	Version control register	0x00F8	R/W

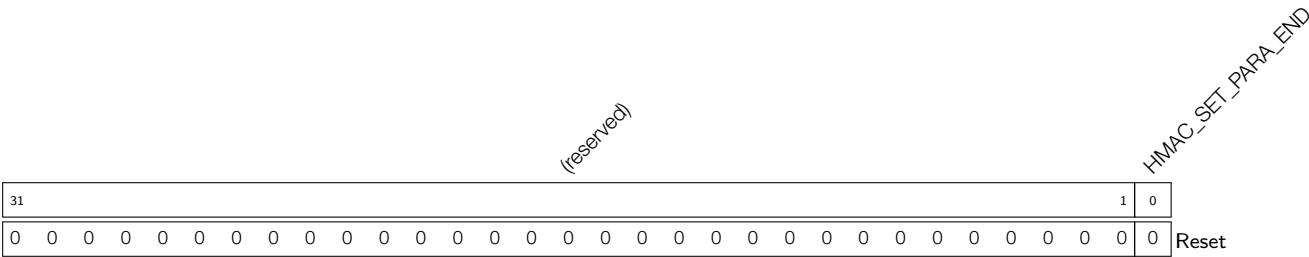
26.7 Registers

Register 26.1: HMAC\_SET\_START\_REG (0x0040)



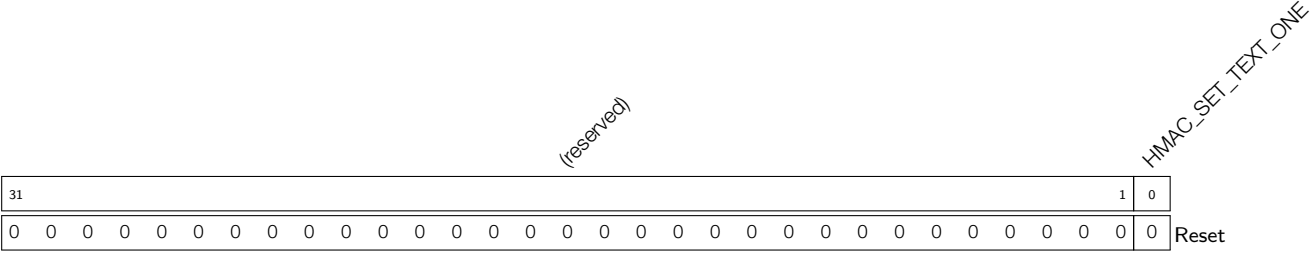
**HMAC\_SET\_START** Set this bit to enable HMAC. (WO)

Register 26.2: HMAC\_SET\_PARA\_FINISH\_REG (0x004C)



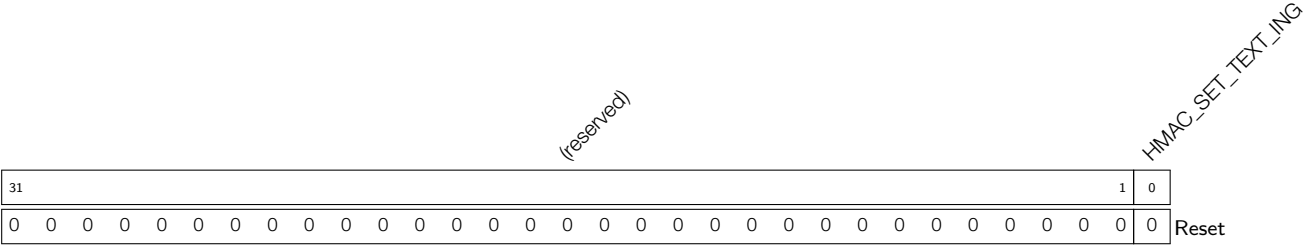
**HMAC\_SET\_PARA\_END** Set this bit to finish HMAC configuration. (WO)

Register 26.3: HMAC\_SET\_MESSAGE\_ONE\_REG (0x0050)



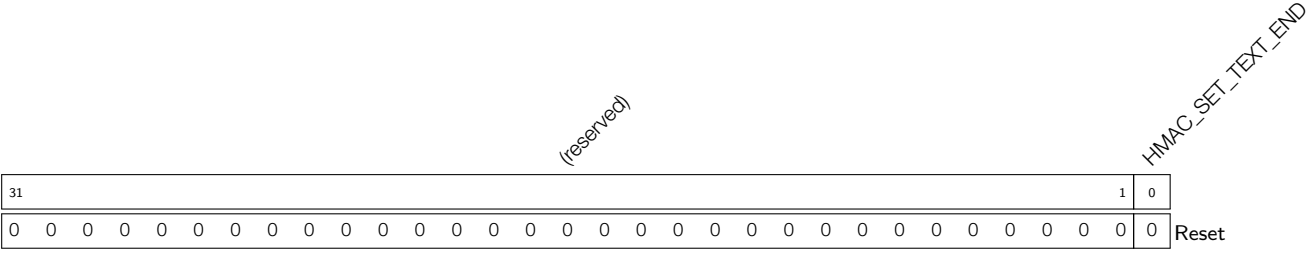
**HMAC\_SET\_TEXT\_ONE** Call SHA to calculate one message block. (WO)

Register 26.4: HMAC\_SET\_MESSAGE\_ING\_REG (0x0054)



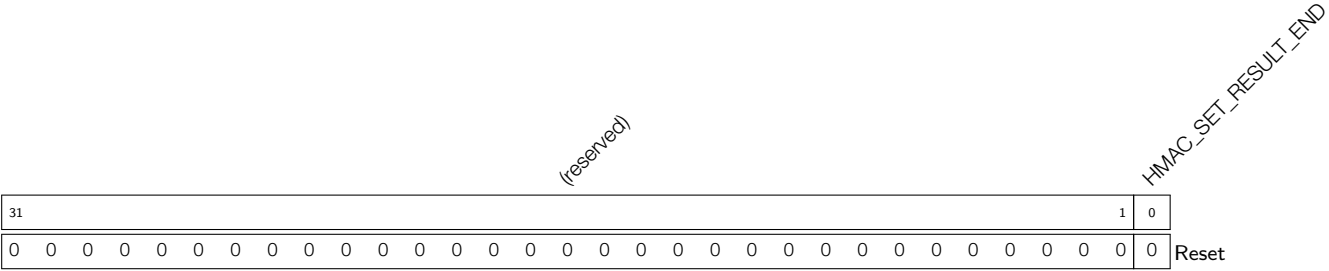
**HMAC\_SET\_TEXT\_ING** Set this bit to show there are still some message blocks to be processed. (WO)

Register 26.5: HMAC\_SET\_MESSAGE\_END\_REG (0x0058)



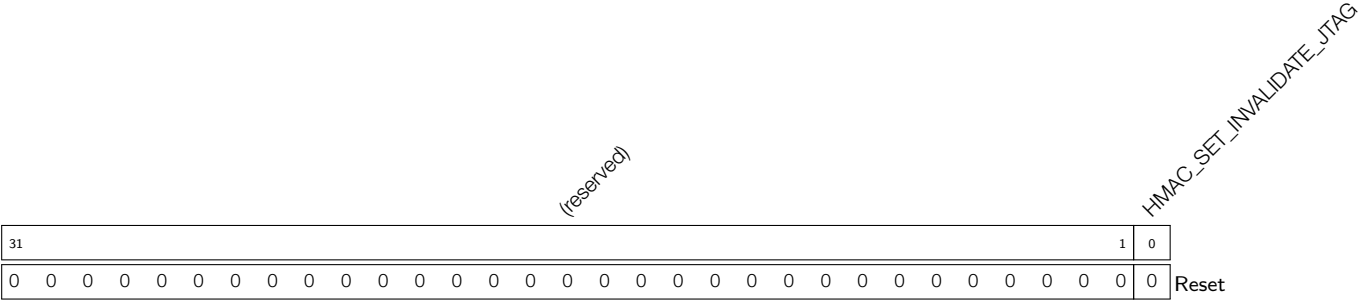
**HMAC\_SET\_TEXT\_END** Set this bit to start hardware padding. (WO)

Register 26.6: HMAC\_SET\_RESULT\_FINISH\_REG (0x005C)



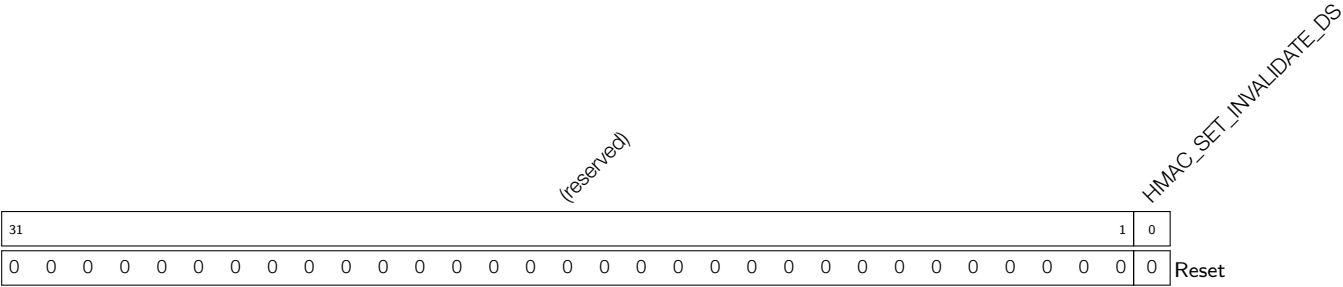
**HMAC\_SET\_RESULT\_END** Set this bit to end upstream and clear the calculation result. (WO)

Register 26.7: HMAC\_SET\_INVALIDATE\_JTAG\_REG (0x0060)



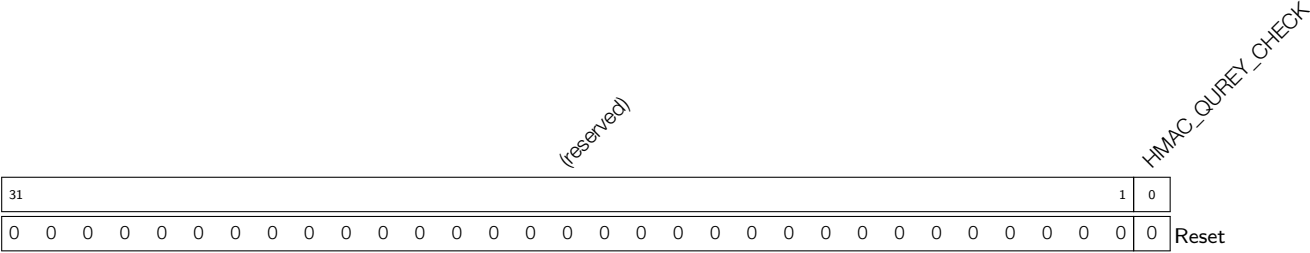
**HMAC\_SET\_INVALIDATE\_JTAG** Set this bit to clear calculation results in JTAG re-enable function under downstream mode. (WO)

Register 26.8: HMAC\_SET\_INVALIDATE\_DS\_REG (0x0064)



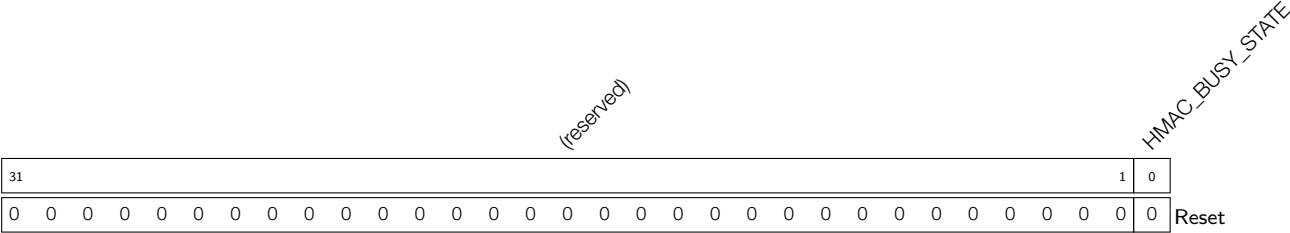
**HMAC\_SET\_INVALIDATE\_DS** Set this bit to clear calculation results in DS function under downstream mode. (WO)

Register 26.9: HMAC\_QUERY\_ERROR\_REG (0x0068)



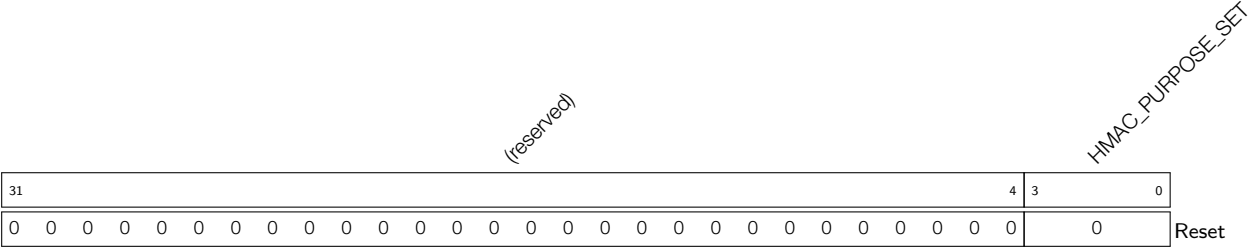
**HMAC\_QUREY\_CHECK** HMAC error status. 0: HMAC key and purpose match. 1: error. (RO)

Register 26.10: HMAC\_QUERY\_BUSY\_REG (0x006C)



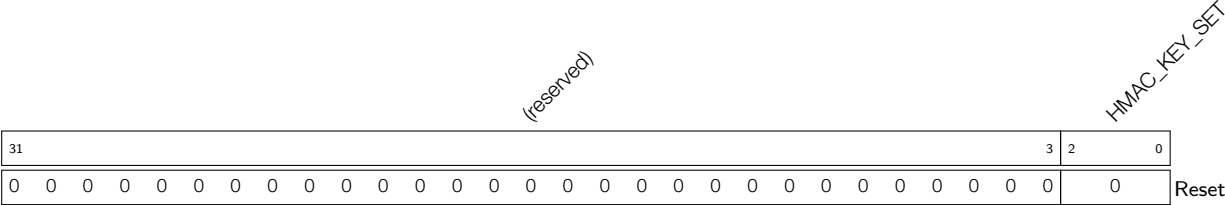
**HMAC\_BUSY\_STATE** The state of HMAC. 1'b0: idle. 1'b1: busy. (RO)

Register 26.11: HMAC\_SET\_PARA\_PURPOSE\_REG (0x0044)



**HMAC\_PURPOSE\_SET** Set HMAC purpose. (WO)

Register 26.12: HMAC\_SET\_PARA\_KEY\_REG (0x0048)



**HMAC\_KEY\_SET** Select HMAC key. (WO)

[Submit Documentation Feedback](#)



[Submit Documentation Feedback](#)



[Submit Documentation Feedback](#)



[Submit Documentation Feedback](#)



Register 26.17: HMAC\_DATE\_REG (0x00F8)

(reserved)		HMAC_DATE	
31	30	29	0
0	0	0x20190402	Reset

**HMAC\_DATE** Store HMAC version information. (R/W)



## 27. On-Chip Sensor and Analog Signal Processing

### 27.1 Overview

ESP32-S2 provides the following on-chip sensor and signal processing peripherals:

- One temperature sensor for measuring the internal temperature of the ESP32-S2 chip.
- Two 13-bit Successive Approximation ADCs (SAR ADCs) controlled by five dedicated controllers that can input analog signals from total of 20 channels. The SAR ADCs can operate in a high-performance mode or a low-power mode.
- Two 8-bit independent DACs to generate analog signals. The DACs also support a cosine wave output mode.

### 27.2 SAR ADCs

#### 27.2.1 Overview

ESP32-S2 integrates two 13-bit SAR ADCs, which are able to measure analog signals from up to 20 analog pads. It is also possible to measure internal signals, such as vdd33. The SAR ADCs are managed by five dedicated controllers:

- Two digital controllers: DIG ADC1 CTRL and DIG ADC2 CTRL, designed for high-performance multi-channel scanning and DMA continuous conversion.
- Two RTC controllers: RTC ADC1 CTRL and RTC ADC2 CTRL, designed for single conversion mode and low power mode.
- One internal controller: PWDET/PKDET CTRL, for power and peak detection (only used by Wi-Fi).

A diagram of the SAR ADCs is shown in Figure 27-1.

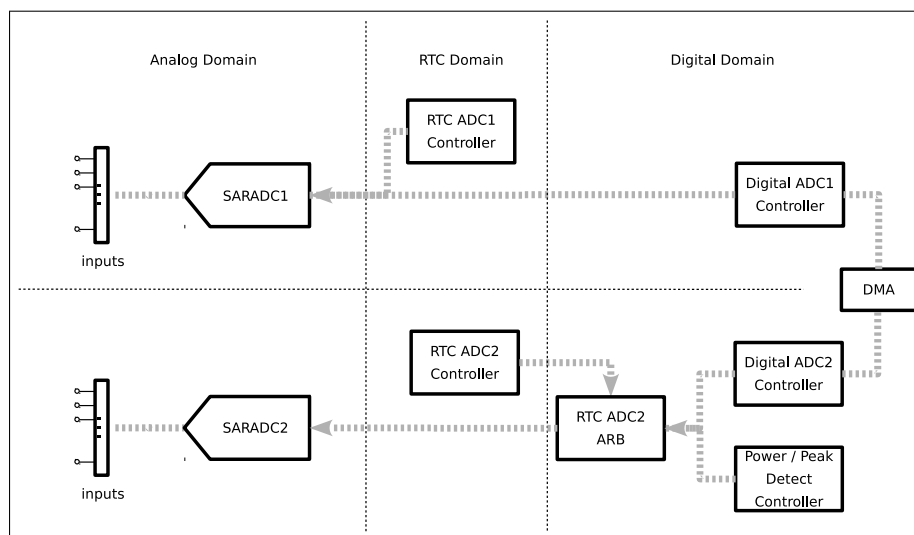


Figure 27-1. SAR ADC Overview

### 27.2.2 Features

- Two SAR ADCs that can operate separately (i.e., convert data simultaneously)
- 12-bit or 13-bit sampling resolution
  - RTC ADC: 13-bit sampling resolution at most
  - DIG ADC: 12-bit sampling resolution at most
- Up to 20 analog input pads
- Configurable channel-scanning duration
- Two RTC ADC controllers, with the following features:
  - Single conversion mode
  - Operation in Deep-sleep mode
  - Can be controlled by ULP coprocessor
- Two DIG ADC controllers, with the following features:
  - Sampled data is transferred using DMA
  - Single conversion mode and continuous conversion mode
  - Multiple channel-scanning mode
  - User-defined channel-scanning sequence
  - Hardware IIR filter with configurable filter coefficient
  - Threshold monitoring. An interrupt will be triggered when the value is greater or less than the threshold.
- One PWDET/PKDET controller only for Wi-Fi internal use to:
  - Monitor power
  - Monitor internal voltage

The differences between the five ADC controllers are summarized in Table 140.

**Table 140: SAR ADC Controllers**

Feature	RTC ADC1	RTC ADC2	DIG ADC1	DIG ADC2	PWDET
Control DAC	Y	-	-	-	-
Support Deep-sleep mode	Y	Y	-	-	-
Controlled by ULP coprocessor	Y	Y	-	-	-
Support vdd33 detection	-	Y	-	Y	-
Support PWDET/PKDET detection	-	-	-	-	Y
Support DMA continuous conversion	-	-	Y	Y	-
Support 13-bit sampling resolution	Y	Y	-	-	-

### 27.2.3 Functional Description

The major components of SAR ADCs and their interconnections are shown in Figure 27-2.

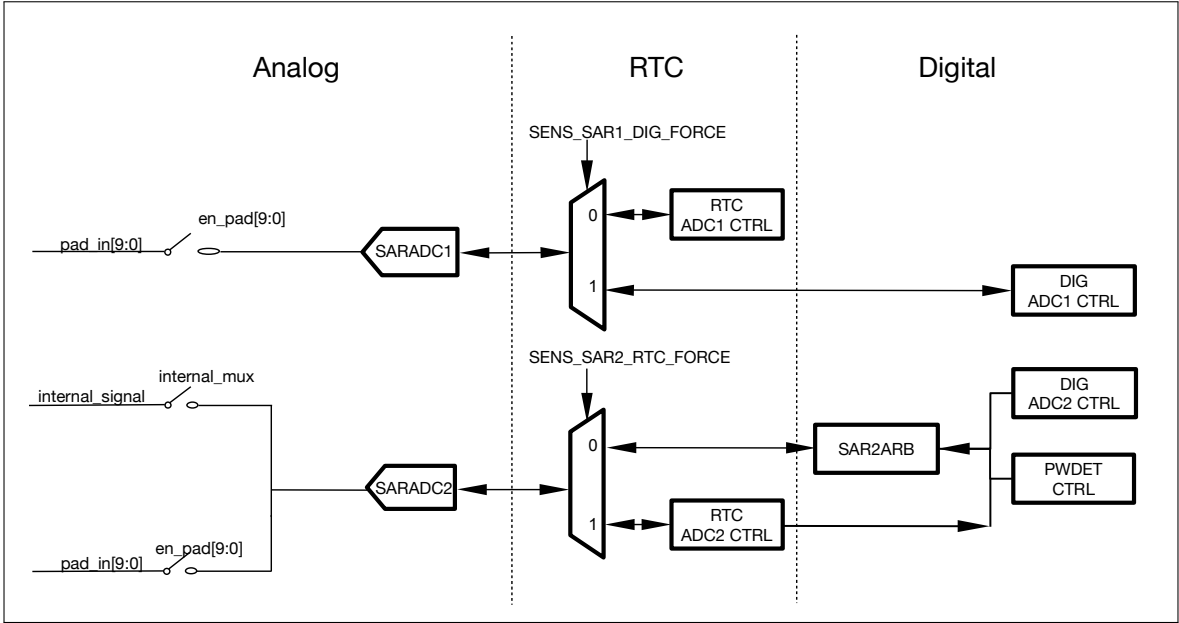


Figure 27-2. SAR ADC Function Overview

### 27.2.3.1 Input Signals

In order to sample an analog signal, an SAR ADC must first select the analog pad or internal signal to measure via an internal multiplexer. A summary of all the analog signals that may be sent to the SAR ADC module for processing by either ADC1 or ADC2 are presented in Table 141.

Table 141: SAR ADC Input Signals

Signal Name	Channel	Processed by
GPIO1	0	SAR ADC1
GPIO2	1	
GPIO3	2	
GPIO4	3	
GPIO5	4	
GPIO6	5	
GPIO7	6	
GPIO8	7	
GPIO9	8	
GPIO10	9	
GPIO11	0	SAR ADC2
GPIO12	1	
GPIO13	2	
GPIO14	3	
GPIO15	4	
GPIO16	5	
GPIO17	6	
GPIO18	7	
GPIO19	8	
GPIO20	9	

Signal Name	Channel	Processed by
pa_pkdet1	n/a	
pa_pkdet2	n/a	
vdd33	n/a	

### 27.2.3.2 ADC Conversion and Attenuation

When the SAR ADCs convert (sample) an analog voltage, the resolution (12-bit or 13-bit) of the conversion spans voltage range from 0 mV to  $V_{ref}$  which is the SAR ADCs internal reference voltage. Thus the output value of the conversion (*data*) will map to analog voltage  $V_{data}$  using the following formulas:

- For 12-bit conversion:  $V_{data} = \frac{V_{ref}}{4095} \times data$
- For 13-bit conversion:  $V_{data} = \frac{V_{ref}}{8191} \times data$

In order to convert voltages larger than  $V_{ref}$ , input signals can be attenuated before being input into the SAR ADCs. The attenuation can be configured to 0 dB, 2.5 dB, 6 dB, and 11 dB.

### 27.2.4 RTC ADC Controllers

The RTC ADC controllers (RTC ADC1 CTRL and RTC ADC2 CTRL) are powered in the RTC power domain, thus allow the SAR ADCs to conduct measurements at a low frequency with minimal power consumption. The outline of a single controller's function is shown in Figure 27-3.

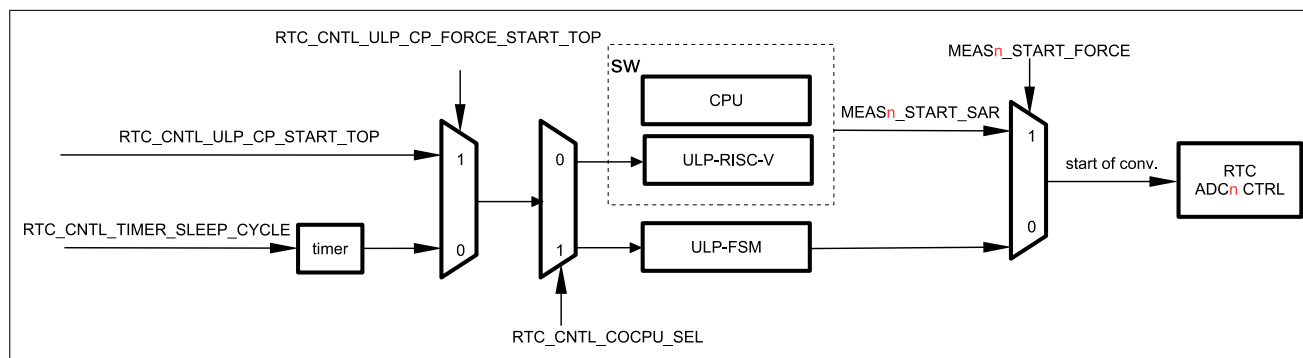


Figure 27-3. RTC SAR ADC Outline

The RTC ADC controllers can be triggered to start a conversion by software or by the ULP coprocessor (ULP-FSM). `SENS_MEASn_START_FORCE` will select whether an RTC ADC controller is triggered by software or by the ULP-FSM.

To trigger the RTC ADC controller to start a conversion by software, set `SENS_MEASn_START_SAR`. When the conversion is complete, `SENS_MEASn_DONE_SAR` will be set and the conversion result will be stored in `SENS_MEASn_DATA_SAR`.

The RTC ADC controllers are intertwined with the ULP coprocessor, as the ULP coprocessor has a built-in instruction to start an ADC conversion. In many cases, the controllers need to cooperate with the ULP coprocessor, e.g.:

- When the controllers periodically monitor a channel during Deep-sleep, ULP coprocessor is the only source to trigger ADC sampling by configuring RTC registers.

- Continuous scanning or DMA is not supported by the controllers. However, it is possible with the help of ULP coprocessor to scan channels continuously in a sequence.

### 27.2.5 DIG ADC Controllers

Compared to the RTC ADC controllers, the DIG ADC controllers are optimized for performance and throughput. The main features of the DIG ADC controllers are outlined below:

- High performance: The clock of the DIG ADC controller is much faster, thus the sample rate is much higher.
- Up to 12-bit sampling resolution
- Support single-channel scanning, double-channel scanning, or alternate-channel scanning mode. The measurement rules for each SAR ADC are defined in pattern tables.
- Scanning can be started by software or by DIG ADC timer, depending on the configuration of [APB\\_SARADC\\_START\\_FORCE](#). Note that DIG ADC timer is enabled by [APB\\_SARADC\\_TIMER\\_EN](#) and its trigger period can be configured by [APB\\_SARADC\\_TIMER\\_TARGET](#).
- Sampled data is transferred to memory via DMA.
- An interrupt will be generated when scanning is finished.

#### 27.2.5.1 Workflow of DIG ADC Controller

**Note:**

The term “start of conversion” is not used in this section, because there is no direct access to starting a single SAR analog-to-digital conversion. Each conversion is started automatically by the DIG ADC controller according to a pattern table. Instead, the term “start of scan” will be used to indicate that the DIG ADC controller will scan a sequence of channels according to a pattern table.

Figure [27-4](#) shows a diagram of the DIG ADC controllers.

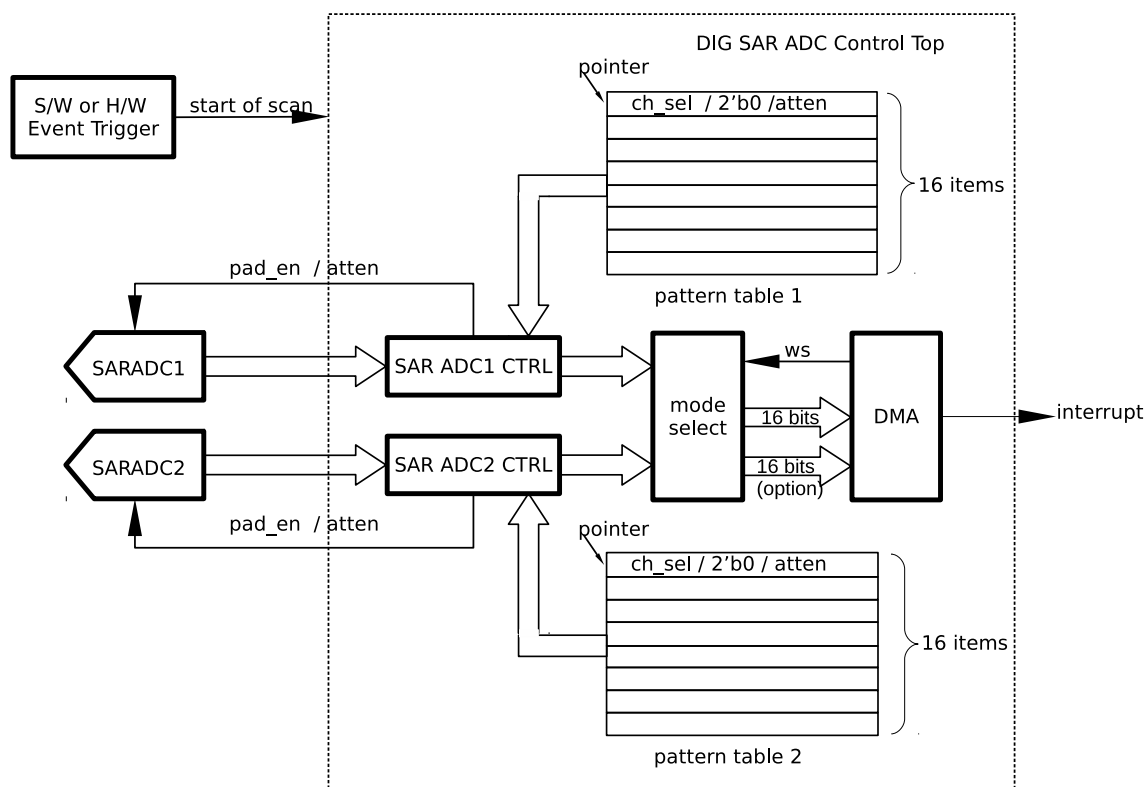


Figure 27-4. Diagram of DIG ADC Controllers

A measurement rule instructs an ADC to sample a channel (and optionally attenuate the signal of the sampled channel). Each of the DIG ADC controllers has a rules table, and each table can store 16 rules. When scanning starts, the controller reads the measurement rules one-by-one from their pattern table. For each controller, a scanning sequence can include 16 different rules at most, before repeating itself.

The 8-bit item (pattern table register) is composed of three fields that contain channel and attenuation information, as shown in Table 142.

Table 142: Fields of Pattern Table Register

Pattern Table Register[7:0]		
ch_sel[3:0]	null	atten[1:0]
Channel	Reserved	Attenuation

The scanning mode configured in `APB_SARADC_WORK_MODE` determines whether the DIG ADC controllers operate completely independently, or in an alternating or synchronized manner. The following scanning modes are supported:

- Single-channel scanning: SAR ADC1 and SAR ADC2 will operate independently according to their own respective pattern tables.
- Double-channel scanning: SAR ADC1 and SAR ADC2 will sample simultaneously (i.e., the two DIG ADC controllers will progress down their own pattern tables in a synchronized manner).
- Alternate-channel scanning: SAR ADC1 and SAR ADC2 will sample alternately (i.e., the two DIG ADC controllers will take turns when progressing down their own pattern tables).

The final data sent to the DMA by each conversion is 16-bit. This data is composed of the 12/11-bit ADC result and some additional information related to the scanning mode:

- For single-channel scanning, 4-bit information on channel selection is added.
- For double-channel scanning or alternate-channel scanning, 4-bit information on channel selection is added plus one extra bit indicating which SAR ADC is the data from.

Two DMA data formats, Type I and Type II, are used to store the data for the above-mentioned two situations, see Table 143 and Table 144.

**Table 143: DMA Data Format (Type I)**

DMA Data Format[15:0]	
ch_sel[3:0]	data[11:0]
Channel	SAR ADC Data

**Table 144: DMA Data Format (Type II)**

DMA Data Format[15:0]		
sar_sel	ch_sel[3:0]	data[10:0]
SAR ADC <sub><i>n</i></sub> ( <i>n</i> = 1 or 2)	Channel	SAR ADC Data

DIG ADCs support 12-bit resolution:

- Type I data format: 12-bit resolution at most.
- Type II data format: 11-bit resolution at most.

### 27.2.5.2 DMA

DIG ADC controllers support direct memory access via the SPI3 DMA and is triggered by DIG ADC dedicated timer. Therefore, the DIG ADC controllers should not be used when SPI3 DMA is already being used. Users can switch the DMA data path to DIG ADC by configuring [APB\\_SARADC\\_APB\\_ADC\\_TRANS](#) via software. For specific DMA configuration, please refer to SPI3 DMA control.

### 27.2.5.3 ADC Filter

The DIG ADC controllers supports automatic filtering of sampled ADC data. The filter's formula is shown below:

$$data_{cur} = \frac{(k-1)data_{prev}}{k} + \frac{data_{in}}{k} - 0.5$$

- $data_{cur}$  is the filtered data value.
- $data_{in}$  is the sampled data value from the ADC.
- $data_{prev}$  is the last filtered data value.
- $k$  is the filter coefficient.

To configure the filter, set the filter coefficient using [APB\\_SARADC\\_ADC \$n\$ \\_FILTER\\_FACTOR](#) then set [APB\\_SARADC\\_ADC \$n\$ \\_FILTER\\_EN](#) to enable the filter. The filter is applied to each DIG ADC controller. However, due to the filter being dependent on previous filtered values ( $data_{prev}$ ), **only one channel on each DIG ADC controller should be scanned when filters are used** so that data of different channels are not mixed by the filter.

#### 27.2.5.4 Threshold Monitoring

Each DIG ADC controller has a threshold monitor that will trigger an interrupt if any of the sampled ADC values exceeds (or falls below) a configured threshold.

- Set the bit [APB\\_SARADC\\_ADC \$n\$ \\_THRES\\_EN](#) to enable threshold monitoring.
- Configure [APB\\_SARADC\\_ADC \$n\$ \\_THRES](#) to set the threshold value.
- Configure [APB\\_SARADC\\_ADC \$n\$ \\_THRES\\_MODE](#) to set the conditions:
  - 0: When  $ADC\_DATA < APB\_SARADC\_ADCn\_THRES$ , a monitor interrupt will be generated.
  - 1: When  $ADC\_DATA \geq APB\_SARADC\_ADCn\_THRES$ , a monitor interrupt will be generated.

Note: Threshold monitoring checks the ADC values on all the channels selected by SAR ADC1 or SAR ADC2, respectively.

#### 27.2.6 SAR ADC2 Arbiter

SAR ADC2 can be controlled by three controllers, namely RTC ADC2 CTRL, DIG ADC2 CTRL, and PWDET/PKDET CTRL. To avoid any possible conflicts and to improve the efficiency of SAR ADC2, ESP32-S2 provides access arbitration for SAR ADC2. The arbiter supports fair arbitration and fixed priority arbitration.

- Fair arbitration mode (cyclic priority arbitration) can be enabled by clearing [APB\\_SARADC\\_ADC\\_ARB\\_FIX\\_PRIORITY](#) in register [APB\\_SARADC\\_ARB\\_CTRL\\_REG](#).
- In fixed priority arbitration, users can set the bits in register [APB\\_SARADC\\_ARB\\_CTRL\\_REG](#), including [APB\\_SARADC\\_ADC\\_ARB\\_RTC\\_PRIORITY](#) (for RTC ADC2 CTRL), [APB\\_SARADC\\_ADC\\_ARB\\_APB\\_PRIORITY](#) (for DIG ADC2 CTRL), and [APB\\_SARADC\\_ADC\\_ARB\\_WIFI\\_PRIORITY](#) (for PWDET CTRL), to configure the priorities for these controllers. A larger value indicates a higher priority.

The arbiter ensures that a higher priority controller can always start a conversion (sample) when required, regardless of whether a lower priority controller already has a conversion in progress. If a higher priority controller starts a conversion whilst the ADC already has a conversion in progress from a lower priority controller, the conversion in progress will be interrupted (stopped). The higher priority controller will then start its conversion.

If a lower priority controller attempts to start a conversion whilst the ADC has a conversion in progress from a higher priority controller, the lower priority controller will not start the conversion.

The value returned by a conversion that is interrupted or not started will be invalid. Therefore, certain data flags are embedded into the output data value to indicate whether the conversion is valid or not.

Note:

- The data flag for RTC ADC2 CTRL is the two higher bits of [SENS\\_MEAS2\\_DATA\\_SAR](#).
  - 2'b10: Conversion is interrupted.



- 2'b01: Conversion is not started.
- 2'b00: The data is valid.
- The data flag for DIG ADC2 CTRL is the ch\_sel[3:0] bits in DMA data.
  - 4'b1111: Conversion is interrupted.
  - 4'b1110: Conversion is not started.
  - Corresponding channel No.: The data is valid.
- The data flag for PWDET/PKDET CTRL is the two higher bits of the sampling result.
  - 2'b10: Conversion is interrupted.
  - 2'b01: Conversion is not started.
  - 2'b00: The data is valid.

Users can configure [APB\\_SARADC\\_ADC\\_ARB\\_GRANT\\_FORCE](#) to mask the arbiter, and set the bits [APB\\_SARADC\\_ADC\\_ARB\\_RTC\\_FORCE](#), [APB\\_SARADC\\_ADC\\_ARB\\_WIFI\\_FORCE](#), and [APB\\_SARADC\\_ADC\\_ARB\\_APB\\_FORCE](#) to authorize corresponding controllers.

Note:

- When the arbiter is masked, only one of the above [APB\\_SARADC\\_ADC\\_ARB\\_XXX\\_FORCE](#) bits can be set to 1.
- The arbiter uses APB\_CLK as its clock source. When the clock frequency is 8 MHz or lower, the arbiter must be masked.
- In sleep mode, the [SENS\\_SAR2\\_RTC\\_FORCE](#) in register [SENS\\_SAR\\_MEAS2\\_MUX\\_REG](#) should be set to 1, masking the arbiter and all the signals from controllers except the RTC controllers.

## 27.3 DACs

### 27.3.1 Overview

ESP32-S2 comes with two built-in 8-bit DACs used to convert digital values into analog output signals (up to two of them). The DACs also support to output cosine waves.

### 27.3.2 Features

- Two 8-bit DAC converters
- Independent or synchronous conversion in both channels
- Voltage reference from VDD3P3\_RTC\_IO pin
- Cosine waves output
- DMA capability
- Fully controlled by ULP coprocessor via registers. See Chapter [28 ULP Coprocessor](#).

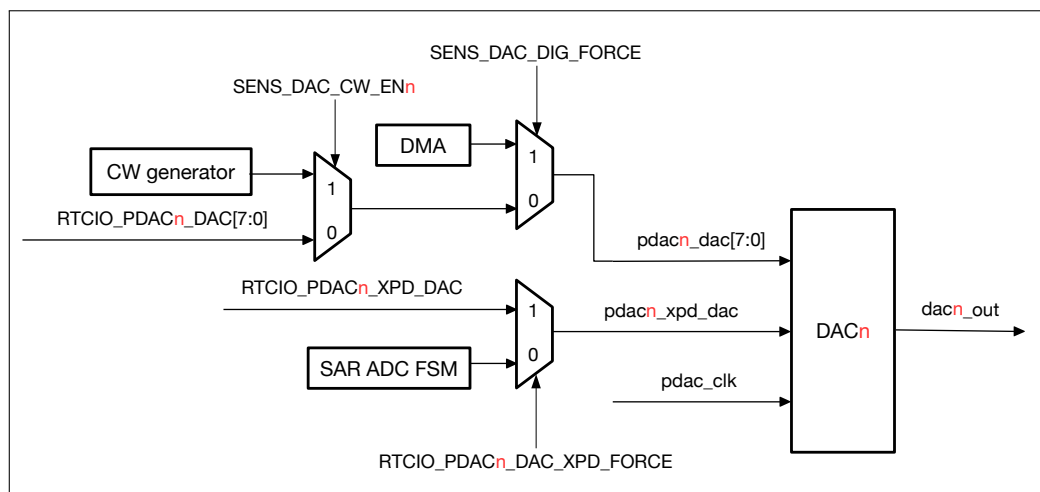


Figure 27-5. Diagram of DAC Function

### 27.3.3 DAC Conversion

The two 8-bit DAC channels can be configured independently. For each DAC channel, the output analog voltage can be calculated as follows:

$$\text{DAC}_n\text{\_OUT} = \text{VDD3P3\_RTC\_IO} \cdot \text{PDAC}_n\text{\_DAC} / 255$$

- VDD3P3\_RTC\_IO is the voltage on pin VDD3P3\_RTC\_IO (usually 3.3 V).
- PDAC<sub>n</sub>\_DAC carries the digital value to be converted into an analog voltage and comes from multiple sources: cosine waveform generator, the register RTCIO\_PAD\_DAC<sub>n</sub>\_REG, or DMA.

The conversion can be started by the register RTCIO\_PDAC<sub>n</sub>\_XPD\_DAC, and controlled by the software or SAR ADC FSM. For more information, please see Figure 27-5.

### 27.3.4 Cosine Wave Generator

The cosine wave (CW) generator can be used to generate a cosine or sine tone. Figure 27-6 shows The function of CW generator.

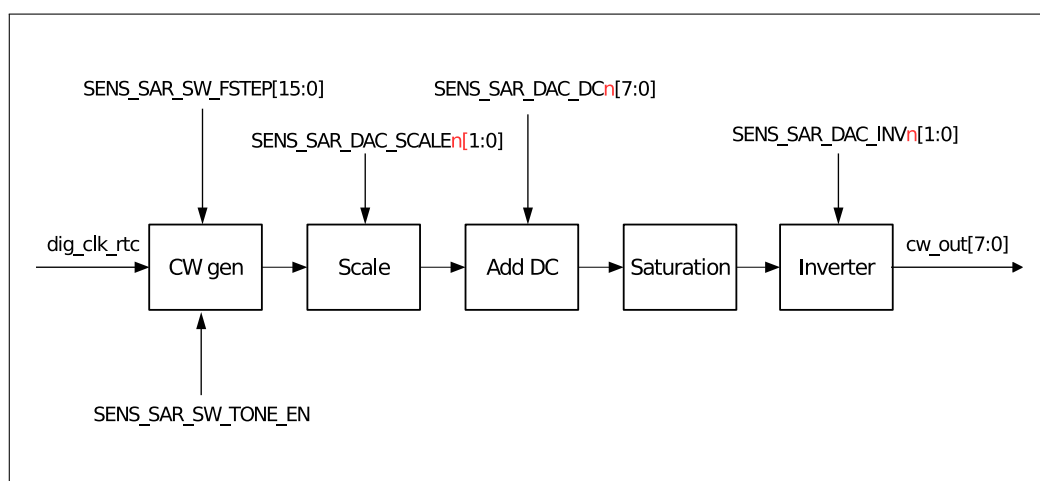


Figure 27-6. Workflow of CW Generator

CW generator features:

- Adjustable frequency

The frequency of CW can be adjusted by register [SENS\\_SW\\_FSTEP](#):

$$\text{freq} = \text{dig\_clk\_rtc\_freq} \cdot \text{SENS\_SW\_FSTEP} / 65535$$

Typically, the frequency of dig\_clk\_rtc is 8 MHz.

- Scaling

The waveform amplitude can be scaled by 1, 1/2, 1/4, or 1/8 by configuring the register [SENS\\_DAC\\_SCALE<sub>n</sub>](#).

- DC offset

The register [SENS\\_DAC\\_DC<sub>n</sub>](#) may introduce DC offset, leading to a saturated result.

- Phase shift

A phase-shift of 0° or 180° can be added by setting register [SENS\\_DAC\\_INV<sub>n</sub>](#).

- 2: 0°
- 3: 180°

### 27.3.5 DMA Support

A DMA controller with dual DAC channels can be used to set the output of two DAC channels. By configuring the register [SENS\\_DAC\\_DIG\\_FORCE](#) and the bit [APB\\_SARADC\\_APB\\_DAC\\_TRANS](#) in the register [APB\\_SARADC\\_APB\\_DAC\\_CTRL\\_REG](#), users can connect DIG\_SARADC\_CLK to DAC clk, and SPI3\_DMA\_OUT to DAC\_DATA for direct memory access.

For details, please refer to Chapter 7 [DMA Controller](#).

## 27.4 Temperature Sensor

### 27.4.1 Overview

ESP32-S2 provides a temperature sensor to monitor temperature changes in real time.

### 27.4.2 Features

- Monitored in real time by ULP coprocessor when in low-power mode
- Triggered by software or by ULP coprocessor
- Configurable temperature offset based on the environment, to improve the accuracy
- Adjustable measurement range

### 27.4.3 Operation Sequence

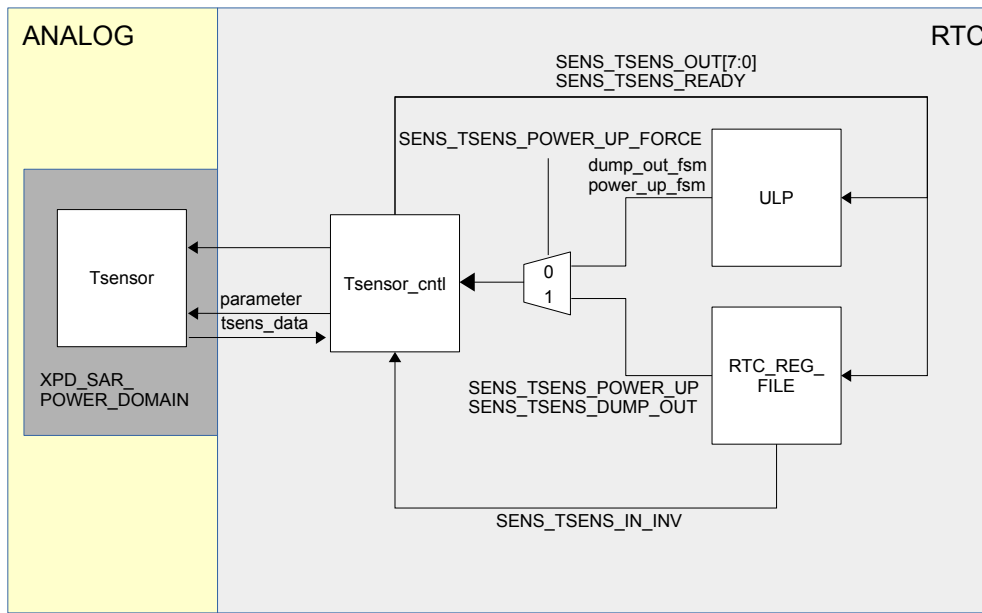


Figure 27-7. Structure of Temperature Sensor

As shown in Figure 27-7, the temperature sensor can be started by software or by ULP coprocessor:

- Started by software, i.e. by CPU or ULP-RISC-V configuring related registers:
  - Start the temperature sensor by setting the registers `SENS_TSENS_POWER_UP_FORCE` and `SENS_TSENS_POWER_UP`.
  - Wait for a while and then configure the register `SENS_TSENS_DUMP_OUT`. The output value gradually approaches the actual temperature linearly as the measurement time increases.
  - Wait for `SENS_TSENS_READY` and read the conversion result from `SENS_TSENS_OUT`.
- Started by ULP-FSM:
  - Clear the register `SENS_TSENS_POWER_UP_FORCE`.
  - ULP-FSM has a built-in instruction for temperature sampling. Executing the instruction can easily complete temperature sampling, see Section 28.5 *ULP-FSM*.

### 27.4.4 Temperature Conversion

The actual temperature can be obtained by converting the output of temperature sensor via the following formula:

$$T(^{\circ}\text{C}) = 0.4386 * \text{VALUE} - 27.88 * \text{offset} - 20.52$$

VALUE in the formula is the output of the temperature sensor, and the offset is determined by the temperature offset `TSENS_DAC`. Users can set I<sup>2</sup>C register `I2C_SARADC_TSENS_ADC` to configure `TSENS_DAC` according to the actual environment (the temperature range).

**Table 145: Temperature Offset**

TSENS_DAC	Temperature Offset	Measurement Range (°C)
5	-2	50 ~ 125
13 or 7	-1	20 ~ 100
15	0	-10 ~ 80
11 or 14	1	-30 ~ 50
10	2	-40 ~ 20

## 27.5 Interrupts

- APB\_SARADC\_ADC1\_THRES\_INT: Triggered when ADC1\_DATA reaches the condition set in [APB\\_SARADC\\_ADC1\\_THRES](#).
- APB\_SARADC\_ADC2\_THRES\_INT: Triggered when ADC2\_DATA reaches the condition set in [APB\\_SARADC\\_ADC2\\_THRES](#).
- APB\_SARADC\_ADC1\_DONE\_INT: Triggered when DIG ADC1 completes data conversion.
- APB\_SARADC\_ADC2\_DONE\_INT: Triggered when DIG ADC2 completes data conversion.

For the interrupts routed to ULP-RISC-V, please refer to Section [28.6.3 ULP-RISC-V Interrupts](#) in Chapter [28 ULP Coprocessor](#).

## 27.6 Base Address

Users can access on-chip sensor, SAR ADCs, and DACs registers with two base addresses, which can be seen in the following table. For more information about accessing peripherals from different buses please see Chapter [1: System and Memory](#).

**Table 146: On-Chip Sensor, SAR ADCs, and DACs Base Addresses**

Module	Bus to Access Peripheral	Base Address
SENSOR (RTC_PERI)	PeriBUS1	0x3F408800
	PeriBUS2	0x60008800
SENSOR (DIGITAL)	PeriBUS1	0x3F440000
	PeriBUS2	0x60040000

Wherein:

- SENSOR (RTC\_PERI) represents the registers, which will be reset due to the power down of RTC\_PERI domain. See Chapter [29 Low-power Management](#).
- SENSOR (DIG\_PERI) represents the registers, which will be reset due to the power down of digital domain. See Chapter [29 Low-power Management](#).

## 27.7 Register Summary

The address in the table is the address offset (relative address) relative to the base address. Please refer to Section [27.6](#) for information about the base address.

### 27.7.1 SENSOR (RTC\_PERI) Register Summary

Name	Description	Address	Access
<b>RTC ADC1 Controller Registers</b>			
<a href="#">SENS_SAR_READER1_CTRL_REG</a>	RTC ADC1 data and sampling control	0x0000	R/W
<a href="#">SENS_SAR_MEAS1_CTRL1_REG</a>	Configure RTC ADC1 controller	0x0008	R/W
<a href="#">SENS_SAR_MEAS1_CTRL2_REG</a>	Control RTC ADC1 conversion and status	0x000C	varies
<a href="#">SENS_SAR_MEAS1_MUX_REG</a>	Select the controller for SAR ADC1	0x0010	R/W
<b>SAR ADC Attention Registers</b>			
<a href="#">SENS_SAR_ATTEN1_REG</a>	Configure SAR ADC1 attenuation	0x0014	R/W
<a href="#">SENS_SAR_ATTEN2_REG</a>	Configure SAR ADC2 attenuation	0x0038	R/W
<b>RTC ADC AMP Control Register</b>			
<a href="#">SENS_SAR_AMP_CTRL3_REG</a>	AMP control register	0x0020	R/W
<b>RTC ADC2 Controller Registers</b>			
<a href="#">SENS_SAR_READER2_CTRL_REG</a>	RTC ADC2 data and sampling control	0x0024	R/W
<a href="#">SENS_SAR_MEAS2_CTRL2_REG</a>	Control RTC ADC2 conversion and status	0x0030	varies
<a href="#">SENS_SAR_MEAS2_MUX_REG</a>	Select the controller for SAR ADC2	0x0034	R/W
<b>Temperature Sensor Registers</b>			
<a href="#">SENS_SAR_TSENS_CTRL_REG</a>	Temperature sensor data control	0x0050	varies
<a href="#">SENS_SAR_TSENS_CTRL2_REG</a>	Temperature sensor control	0x0054	R/W
<b>DAC Registers</b>			
<a href="#">SENS_SAR_DAC_CTRL1_REG</a>	DAC control	0x011C	R/W
<a href="#">SENS_SAR_DAC_CTRL2_REG</a>	DAC output control	0x0120	R/W
<b>IO MUX Clock Gate</b>			
<a href="#">SENS_SAR_IO_MUX_CONF_REG</a>	Configure and reset IO MUX	0x0144	R/W

### 27.7.2 SENSOR (DIG\_PERI) Register Summary

Name	Description	Address	Access
<b>DIG ADC Controller Registers</b>			
<a href="#">APB_SARADC_CTRL_REG</a>	DIG ADC common configuration	0x0000	R/W
<a href="#">APB_SARADC_CTRL2_REG</a>	DIG ADC common configuration	0x0004	R/W
<a href="#">APB_SARADC_CLKM_CONF_REG</a>	Configure DIG ADC clock	0x005C	R/W
<b>DIG ADC1 Pattern Table Registers</b>			
<a href="#">APB_SARADC_SAR1_PATT_TAB1_REG</a>	Item 0 ~ 3 for pattern table 1 (each item one byte)	0x0018	R/W
<a href="#">APB_SARADC_SAR1_PATT_TAB2_REG</a>	Item 4 ~ 7 for pattern table 1 (each item one byte)	0x001C	R/W
<a href="#">APB_SARADC_SAR1_PATT_TAB3_REG</a>	Item 8 ~ 11 for pattern table 1 (each item one byte)	0x0020	R/W
<a href="#">APB_SARADC_SAR1_PATT_TAB4_REG</a>	Item 12 ~ 15 for pattern table 1 (each item one byte)	0x0024	R/W
<b>DIG ADC2 Pattern Table Registers</b>			
<a href="#">APB_SARADC_SAR2_PATT_TAB1_REG</a>	Item 0 ~ 3 for pattern table 2 (each item one byte)	0x0028	R/W
<a href="#">APB_SARADC_SAR2_PATT_TAB2_REG</a>	Item 4 ~ 7 for pattern table 2 (each item one byte)	0x002C	R/W
<a href="#">APB_SARADC_SAR2_PATT_TAB3_REG</a>	Item 8 ~ 11 for pattern table 2 (each item one byte)	0x0030	R/W

Name	Description	Address	Access
<a href="#">APB_SARADC_SAR2_PATT_TAB4_REG</a>	Item 12 ~ 15 for pattern table 2 (each item one byte)	0x0034	R/W
<b>DIG ADC2 Arbiter Register</b>			
<a href="#">APB_SARADC_ARB_CTRL_REG</a>	Configure the settings of DIG ADC2 arbiter	0x0038	R/W
<b>DIG ADC2 Filter Registers</b>			
<a href="#">APB_SARADC_FILTER_CTRL_REG</a>	Configure the settings of DIG ADC2 filter	0x003C	R/W
<a href="#">APB_SARADC_FILTER_STATUS_REG</a>	Data status of DIG ADC2 filter	0x0040	RO
<b>DIG ADC2 Threshold Register</b>			
<a href="#">APB_SARADC_THRES_CTRL_REG</a>	Configure monitor threshold for DIG ADC2	0x0044	R/W
<b>DIG ADC Interrupt Registers</b>			
<a href="#">APB_SARADC_INT_ENA_REG</a>	Enable DIG ADC interrupts	0x0048	R/W
<a href="#">APB_SARADC_INT_RAW_REG</a>	DIG ADC interrupt raw bits	0x004C	RO
<a href="#">APB_SARADC_INT_ST_REG</a>	DIG ADC interrupt status	0x0050	RO
<a href="#">APB_SARADC_INT_CLR_REG</a>	Clear DIG ADC interrupts	0x0054	WO
<b>DMA Register for DIG ADC</b>			
<a href="#">APB_SARADC_DMA_CONF_REG</a>	Configure digital ADC DMA path	0x0058	R/W
<b>DAC Control Register</b>			
<a href="#">APB_SARADC_DAC_CTRL_REG</a>	Configure DAC settings	0x0060	R/W
<b>Version Register</b>			
<a href="#">APB_SARADC_CTRL_DATE_REG</a>	Version control register	0x03FC	R/W

## 27.8 Register

### 27.8.1 SENSOR (RTC\_PERI) Registers

Register 27.1: SENS\_SAR\_READER1\_CTRL\_REG (0x0000)

(reserved)																								(reserved)																								SENS_SAR1_CLK_DIV																							
31		30		29		28		27																		8		7																		0																									
0		0		1		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		2		Reset																							

Reset

**SENS\_SAR1\_CLK\_DIV** Clock divider. (R/W)

**SENS\_SAR1\_DATA\_INV** Invert SAR ADC1 data. (R/W)

**SENS\_SAR1\_INT\_EN** Enable SAR ADC1 to send out interrupt. (R/W)

**Register 27.2: SENS\_SAR\_MEAS1\_CTRL1\_REG (0x0008)**

(reserved)								SENS_RTC_SARADC_CLKGATE_EN SENS_RTC_SARADC_RESET																(reserved)								
31								24	23	22	21																					0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**SENS\_RTC\_SARADC\_RESET** SAR ADC software reset. (R/W)

**SENS\_RTC\_SARADC\_CLKGATE\_EN** Enable bit of SAR ADC clock gate. (R/W)

**Register 27.3: SENS\_SAR\_MEAS1\_CTRL2\_REG (0x000C)**

SENS_SAR1_EN_PAD_FORCE										SENS_SAR1_EN_PAD										SENS_MEAS1_START_FORCE SENS_MEAS1_START_SAR SENS_MEAS1_DONE_SAR										SENS_MEAS1_DATA_SAR									
31	30							19							18	17	16	15														0							
0	0														0	0	0	0														Reset							

**SENS\_MEAS1\_DATA\_SAR** SAR ADC1 data. (RO)

**SENS\_MEAS1\_DONE\_SAR** Indicate SAR ADC1 conversion is done. (RO)

**SENS\_MEAS1\_START\_SAR** SAR ADC1 controller (in RTC) starts conversion, active only when SENS\_MEAS1\_START\_FORCE = 1. (R/W)

**SENS\_MEAS1\_START\_FORCE** 1: SAR ADC1 controller (in RTC) is started by software. 0: SAR ADC1 controller is started by ULP coprocessor. (R/W)

**SENS\_SAR1\_EN\_PAD** SAR ADC1 pad enable bitmap, active only when SENS\_SAR1\_EN\_PAD\_FORCE = 1. (R/W)

**SENS\_SAR1\_EN\_PAD\_FORCE** 1: SAR ADC1 pad enable bitmap is controlled by software. 0: SAR ADC1 pad enable bitmap is controlled by ULP coprocessor. (R/W)





## 586

ESP32-S2 TRM (Preliminary V0.5)

[Submit Documentation Feedback](#)

[illegible]

**SENS\_SAR2\_INT\_EN** Enable SAR ADC2 to send out interrupt. (R/W)

**Register 27.9: SENS\_SAR\_MEAS2\_CTRL2\_REG (0x0030)**

SENS_SAR2_EN_PAD_FORCE		SENS_SAR2_EN_PAD		SENS_MEAS2_START_FORCE		SENS_MEAS2_START_SAR		SENS_MEAS2_DONE_SAR		SENS_MEAS2_DATA_SAR		
31	30	19	18	17	16	15	0					
0	0			0	0	0	0					Reset

**SENS\_MEAS2\_DATA\_SAR** SAR ADC2 data. (RO)

**SENS\_MEAS2\_DONE\_SAR** Indicate SAR ADC2 conversion is done. (RO)

**SENS\_MEAS2\_START\_SAR** SAR ADC2 controller (in RTC) starts conversion, active only when SENS\_MEAS2\_START\_FORCE = 1. (R/W)

**SENS\_MEAS2\_START\_FORCE** 1: SAR ADC2 controller (in RTC) is started by software. 0: SAR ADC2 controller is started by ULP coprocessor. (R/W)

**SENS\_SAR2\_EN\_PAD** SAR ADC2 pad enable bitmap, active only when SENS\_SAR2\_EN\_PAD\_FORCE = 1. (R/W)

**SENS\_SAR2\_EN\_PAD\_FORCE** 1: SAR ADC2 pad enable bitmap is controlled by software. 0: SAR ADC2 pad enable bitmap is controlled by ULP coprocessor. (R/W)

**Register 27.10: SENS\_SAR\_MEAS2\_MUX\_REG (0x0034)**

SENS_SAR2_RTC_FORCE		SENS_SAR2_PWDET_CCT		(reserved)																									
31	30	28	27																										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
																													Reset

**SENS\_SAR2\_PWDET\_CCT** SAR2\_PWDET\_CCT, PA power detector capacitance tuning. (R/W)

**SENS\_SAR2\_RTC\_FORCE** In sleep, force to use RTC to control ADC. (R/W)

**Register 27.11: SENS\_SAR\_TSENS\_CTRL\_REG (0x0050)**

(reserved)								SENS_TSENS_DUMP_OUT SENS_TSENS_POWER_UP_FORCE SENS_TSENS_POWER_UP				SENS_TSENS_CLK_DIV				SENS_TSENS_IN_INV SENS_TSENS_INT_EN				(reserved)				SENS_TSENS_READY				SENS_TSENS_OUT																		
31								25				24	23	22				21				14				13	12	11				9				8	7				0					
0								0				0	0	0				6				0				1	0				0				0				0x0				Reset			

**SENS\_TSENS\_OUT** Temperature sensor data out. (RO)

**SENS\_TSENS\_READY** Indicate temperature sensor out ready. (RO)

**SENS\_TSENS\_INT\_EN** Enable temperature sensor to send out interrupt. (R/W)

**SENS\_TSENS\_IN\_INV** Invert temperature sensor data. (R/W)

**SENS\_TSENS\_CLK\_DIV** Temperature sensor clock divider. (R/W)

**SENS\_TSENS\_POWER\_UP** Temperature sensor power up. (R/W)

**SENS\_TSENS\_POWER\_UP\_FORCE** 1: dump out and power up controlled by software. 0: by FSM.  
(R/W)

**SENS\_TSENS\_DUMP\_OUT** Temperature sensor dump out only active when  
SENS\_TSENS\_POWER\_UP\_FORCE = 1. (R/W)

**Register 27.12: SENS\_SAR\_TSENS\_CTRL2\_REG (0x0054)**

(reserved)												SENS_TSENS_RESET SENS_TSENS_CLKGATE_EN				(reserved)													
31													17	16	15	14													0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**SENS\_TSENS\_CLKGATE\_EN** Enable temperature sensor clock. (R/W)

**SENS\_TSENS\_RESET** Reset temperature sensor. (R/W)

Register 27.13: SENS\_SAR\_DAC\_CTRL1\_REG (0x011C)

(reserved)				SENS_DAC_CLKGATE_EN				SENS_DAC_RESET				SENS_DAC_CLK_INV				SENS_DAC_CLK_FORCE_HIGH				SENS_DAC_CLK_FORCE_LOW				(reserved)				SENS_SW_TONE_EN				SENS_SW_FSTEP																							
31				28	27	26	25	24	23	22	21					17	16	15																					0																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																			Reset																	

**SENS\_SW\_FSTEP** Frequency step for CW generator can be used to adjust the frequency. (R/W)

**SENS\_SW\_TONE\_EN** 0: disable CW generator. 1: enable CW generator. (R/W)

**SENS\_DAC\_DIG\_FORCE** 0: DAC1 and DAC2 do not use DMA. 1: DAC1 and DAC2 use DMA. (R/W)

**SENS\_DAC\_CLK\_FORCE\_LOW** 1: force PDAC\_CLK to low. (R/W)

**SENS\_DAC\_CLK\_FORCE\_HIGH** 1: force PDAC\_CLK to high. (R/W)

**SENS\_DAC\_CLK\_INV** 1: invert PDAC\_CLK. (R/W)

**SENS\_DAC\_RESET** Reset DAC by software. (R/W)

**SENS\_DAC\_CLKGATE\_EN** DAC clock gate enable bit. (R/W)

**Register 27.14: SENS\_SAR\_DAC\_CTRL2\_REG (0x0120)**

(reserved)							SENS_DAC_CW_EN2		SENS_DAC_CW_EN1		SENS_DAC_INV2		SENS_DAC_INV1		SENS_DAC_SCALE2		SENS_DAC_SCALE1		SENS_DAC_DC2				SENS_DAC_DC1									
31						26	25	24	23	22	21	20	19	18	17	16	15						8	7						0		
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0					0					Reset				

**SENS\_DAC\_DC1** DC offset for DAC1 CW generator. (R/W)

**SENS\_DAC\_DC2** DC offset for DAC2 CW generator. (R/W)

**SENS\_DAC\_SCALE1** DAC1 scaling. 00: no scale. 01: scale to 1/2. 10: scale to 1/4. 11: scale to 1/8. (R/W)

**SENS\_DAC\_SCALE2** DAC2 scaling. 00: no scale. 01: scale to 1/2. 10: scale to 1/4. 11: scale to 1/8. (R/W)

**SENS\_DAC\_INV1** Invert DAC1. 00: do not invert any bits. 01: invert all bits. 10: invert MSB. 11: invert all bits except MSB. (R/W)

**SENS\_DAC\_INV2** Invert DAC2. 00: do not invert any bits. 01: invert all bits. 10: invert MSB. 11: invert all bits except MSB. (R/W)

**SENS\_DAC\_CW\_EN1** 1: select CW generator as source for PDAC1\_DAC. 0: select register RT-CIO\_PDAC1\_DAC as source for PDAC1\_DAC. (R/W)

**SENS\_DAC\_CW\_EN2** 1: select CW generator as source for PDAC2\_DAC. 0: select register RT-CIO\_PDAC2\_DAC as source for PDAC2\_DAC. (R/W)

**Register 27.15: SENS\_SAR\_IO\_MUX\_CONF\_REG (0x0144)**

SENS_IOMUX_CLK_GATE_EN		SENS_IOMUX_RESET		(reserved)																										
31	30	29	0																											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**SENS\_IOMUX\_RESET** Reset IO MUX by software (R/W)

**SENS\_IOMUX\_CLK\_GATE\_EN** IO MUX clock gate enable bit (R/W)

## 27.8.2 SENSOR (DIG\_PERI) Registers

Register 27.16: APB\_SARADC\_CTRL\_REG (0x0000)

APB_SARADC_WAIT_APB_CYCLE		APB_SARADC_XPD_SAR_FORCE		APB_SARADC_DATA_SAR_SEL		APB_SARADC_SAR2_PATT_P_CLEAR		APB_SARADC_SAR1_PATT_P_CLEAR		APB_SARADC_SAR2_PATT_LEN		APB_SARADC_SAR1_PATT_LEN		APB_SARADC_SAR_CLK_DIV		APB_SARADC_SAR_CLK_GATED		APB_SARADC_SAR_SEL		APB_SARADC_WORK_MODE		APB_SARADC_START		APB_SARADC_START_FORCE	
(reserved)		(reserved)																							
31	30	29	28	27	26	25	24	23	22	19	18	15	14	7	6	5	4	3	2	1	0				
1	0	0	0	0	0	0	0	15		15		4		1	0	0	0	0	0	0	Reset				

**APB\_SARADC\_START\_FORCE** 0: select FSM to start SAR ADC. 1: select software to start SAR ADC. (R/W)

**APB\_SARADC\_START** Start SAR ADC by software. (R/W)

**APB\_SARADC\_WORK\_MODE** 0: single-channel scan mode. 1: double-channel scan mode. 2: alternate-channel scan mode. (R/W)

**APB\_SARADC\_SAR\_SEL** 0: select SAR ADC1. 1: select SAR ADC2, only work for single-channel scan mode. (R/W)

**APB\_SARADC\_SAR\_CLK\_GATED** SAR clock gate enable bit. (R/W)

**APB\_SARADC\_SAR\_CLK\_DIV** SAR clock divider. (R/W)

**APB\_SARADC\_SAR1\_PATT\_LEN** 0 ~ 15 means length 1 ~ 16. (R/W)

**APB\_SARADC\_SAR2\_PATT\_LEN** 0 ~ 15 means length 1 ~ 16. (R/W)

**APB\_SARADC\_SAR1\_PATT\_P\_CLEAR** Clear the pointer of pattern table for DIG ADC1 CTRL. (R/W)

**APB\_SARADC\_SAR2\_PATT\_P\_CLEAR** Clear the pointer of pattern table for DIG ADC2 CTRL. (R/W)

**APB\_SARADC\_DATA\_SAR\_SEL** 1: sar\_sel will be coded to the MSB of the 16-bit output data, in this case the resolution should not be larger than 11 bits. (R/W)

**APB\_SARADC\_XPD\_SAR\_FORCE** Force option to xpd sar blocks. (R/W)

**APB\_SARADC\_WAIT\_ARB\_CYCLE** Wait arbit signal stable after sar\_done. (R/W)

Register 27.17: APB\_SARADC\_CTRL2\_REG (0x0004)

(reserved)								APB_SARADC_TIMER_EN								APB_SARADC_TIMER_TARGET								(reserved)				APB_SARADC_SAR2_INV				APB_SARADC_SAR1_INV				APB_SARADC_MAX_MEAS_NUM								APB_SARADC_MEAS_NUM_LIMIT																																																																				
31								25								24	23								12								11	10	9	8								1								0																																																												
0								0								0								0								0								0								0								0								10								0								0								0								255								0								Reset

**APB\_SARADC\_MEAS\_NUM\_LIMIT** Enable limit times of SAR ADC sample. (R/W)

**APB\_SARADC\_MAX\_MEAS\_NUM** Set maximum conversion number. (R/W)

**APB\_SARADC\_SAR1\_INV** 1: data to DIG ADC1 CTRL is inverted, otherwise not. (R/W)

**APB\_SARADC\_SAR2\_INV** 1: data to DIG ADC2 CTRL is inverted, otherwise not. (R/W)

**APB\_SARADC\_TIMER\_TARGET** Set SAR ADC timer target. (R/W)

**APB\_SARADC\_TIMER\_EN** Enable SAR ADC timer trigger. (R/W)

Register 27.18: APB\_SARADC\_CLKM\_CONF\_REG (0x005C)

(reserved)										APB_SARADC_CLK_SEL										(reserved)										APB_SARADC_CLKM_DIV_A										APB_SARADC_CLKM_DIV_B										APB_SARADC_CLKM_DIV_NUM									
31										23										22	21	20	19	14										13	8										7	0													
0										0										0	0	0x0										0x0										4										Reset							

**APB\_SARADC\_CLKM\_DIV\_NUM** Integral DIG\_ADC clock divider value (R/W)

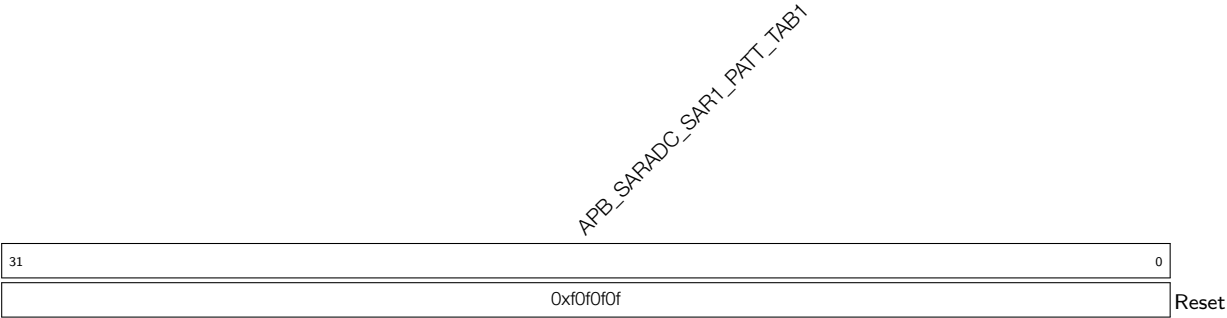
**APB\_SARADC\_CLKM\_DIV\_B** Fractional clock divider numerator value (R/W)

**APB\_SARADC\_CLKM\_DIV\_A** Fractional clock divider denominator value (R/W)

**APB\_SARADC\_CLK\_SEL** 1: select APLL. 2: select APB\_CLK. Other values: disable clock. (R/W)

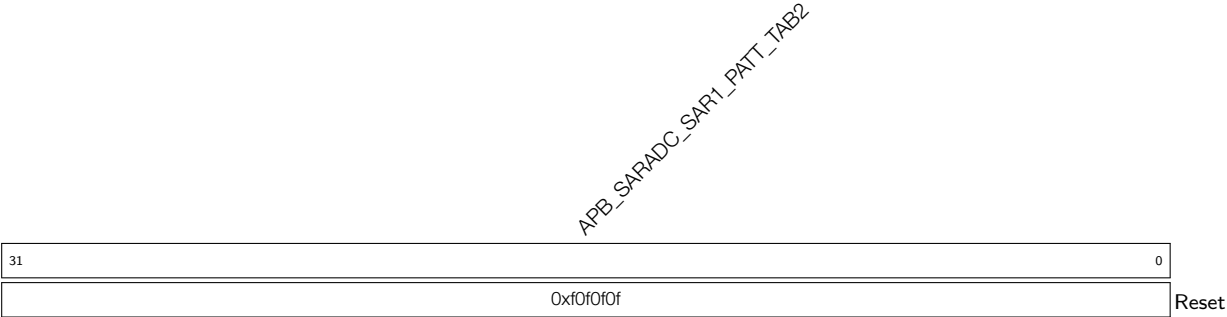


Register 27.19: APB\_SARADC\_SAR1\_PATT\_TAB1\_REG (0x0018)



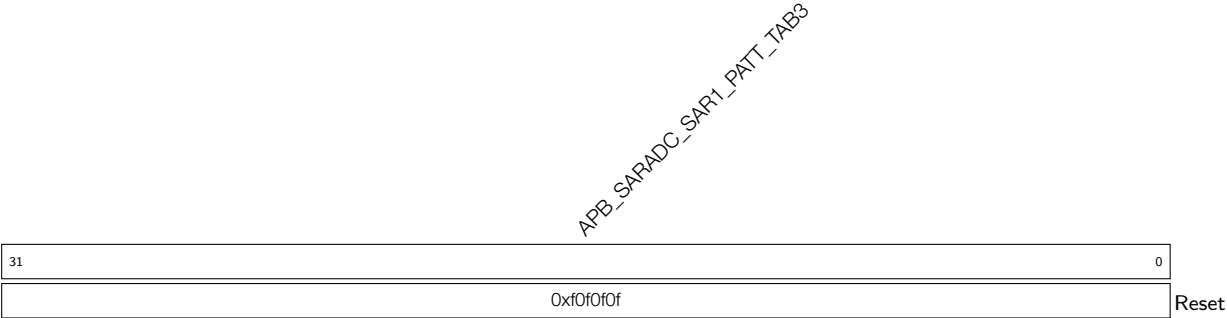
**APB\_SARADC\_SAR1\_PATT\_TAB1** Item 0 ~ 3 for pattern table 1 (each item one byte). (R/W)

Register 27.20: APB\_SARADC\_SAR1\_PATT\_TAB2\_REG (0x001C)



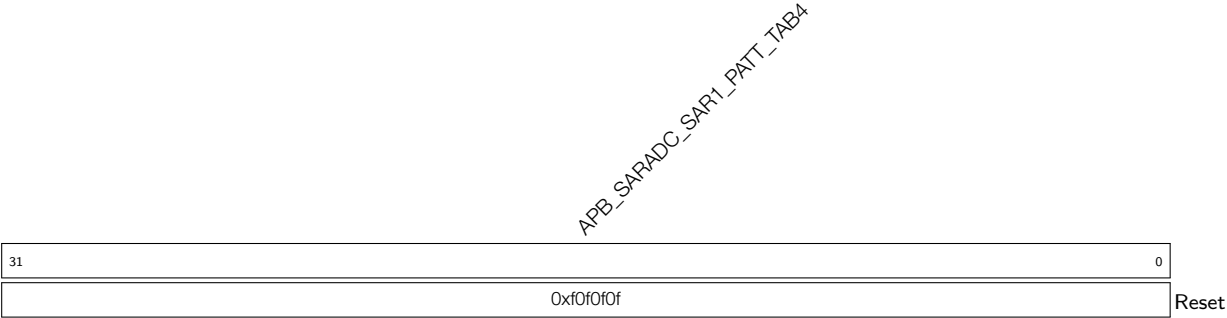
**APB\_SARADC\_SAR1\_PATT\_TAB2** Item 4 ~ 7 for pattern table 1 (each item one byte). (R/W)

Register 27.21: APB\_SARADC\_SAR1\_PATT\_TAB3\_REG (0x0020)



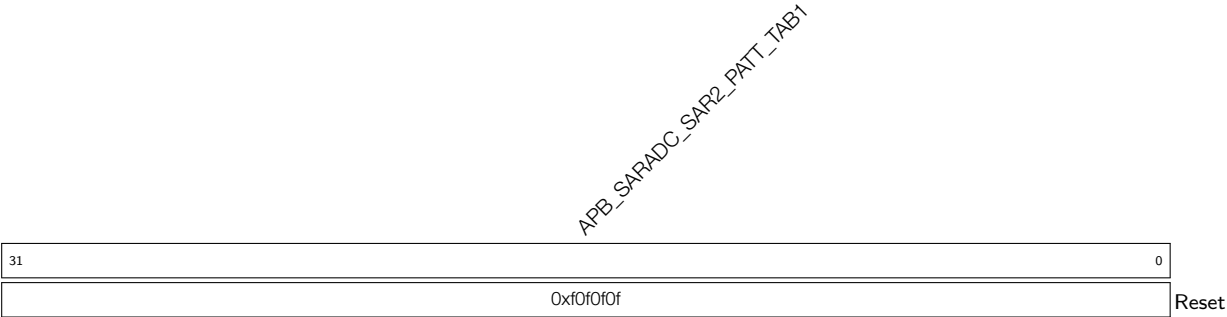
**APB\_SARADC\_SAR1\_PATT\_TAB3** Item 8 ~ 11 for pattern table 1 (each item one byte). (R/W)

Register 27.22: APB\_SARADC\_SAR1\_PATT\_TAB4\_REG (0x0024)



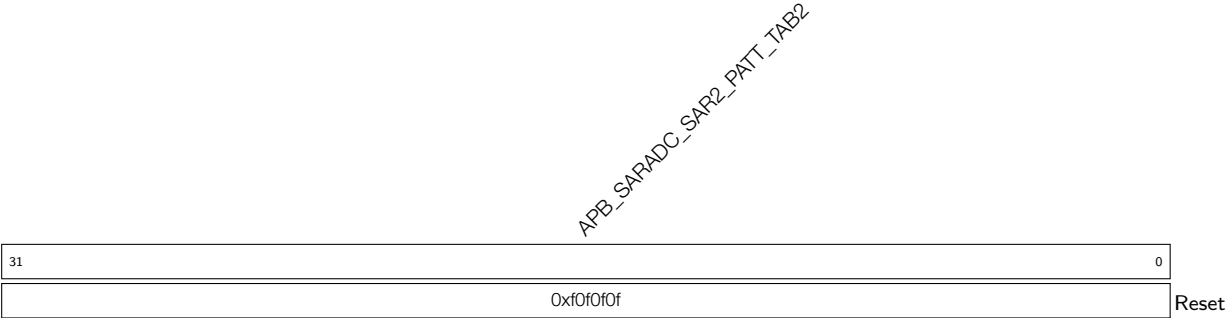
APB\_SARADC\_SAR1\_PATT\_TAB4 Item 12 ~ 15 for pattern table 1 (each item one byte). (R/W)

Register 27.23: APB\_SARADC\_SAR2\_PATT\_TAB1\_REG (0x0028)



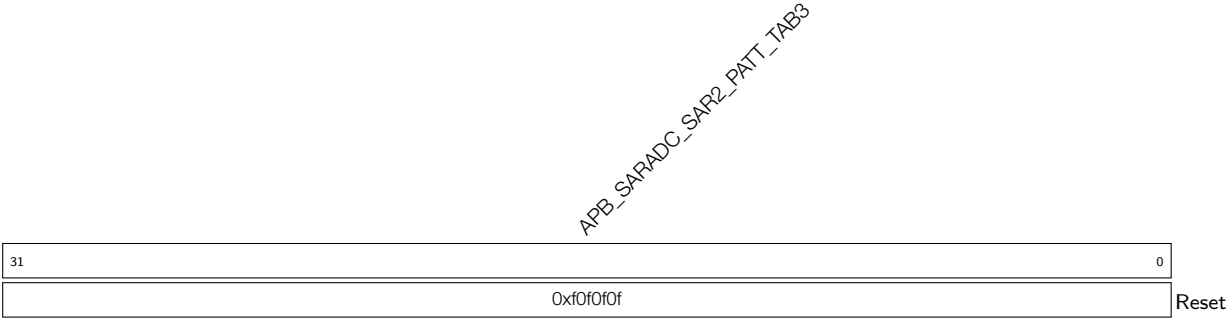
APB\_SARADC\_SAR2\_PATT\_TAB1 Item 0 ~ 3 for pattern table 2 (each item one byte). (R/W)

Register 27.24: APB\_SARADC\_SAR2\_PATT\_TAB2\_REG (0x002C)



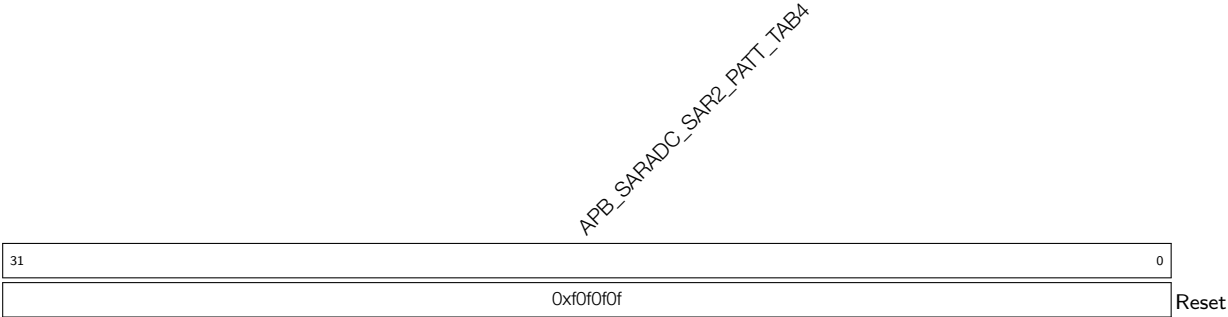
APB\_SARADC\_SAR2\_PATT\_TAB2 Item 4 ~ 7 for pattern table 2 (each item one byte). (R/W)

Register 27.25: APB\_SARADC\_SAR2\_PATT\_TAB3\_REG (0x0030)



**APB\_SARADC\_SAR2\_PATT\_TAB3** Item 8 ~ 11 for pattern table 2 (each item one byte). (R/W)

Register 27.26: APB\_SARADC\_SAR2\_PATT\_TAB4\_REG (0x0034)



**APB\_SARADC\_SAR2\_PATT\_TAB4** Item 12 ~ 15 for pattern table 2 (each item one byte). (R/W)

## Register 27.27: APB\_SARADC\_ARB\_CTRL\_REG (0x0038)

(reserved)																				APB_SARADC_ADC_ARB_FIX_PRIORITY APB_SARADC_ADC_ARB_WIFI_PRIORITY APB_SARADC_ADC_ARB_RTC_PRIORITY APB_SARADC_ADC_ARB_APB_PRIORITY APB_SARADC_ADC_ARB_GRANT_FORCE APB_SARADC_ADC_ARB_WIFI_FORCE APB_SARADC_ADC_ARB_RTC_FORCE APB_SARADC_ADC_ARB_APB_FORCE (reserved)							
31													13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0		

**APB\_SARADC\_ADC\_ARB\_APB\_FORCE** ADC2 arbiter forces to enable DIG ADC2 CTRL. (R/W)

**APB\_SARADC\_ADC\_ARB\_RTC\_FORCE** ADC2 arbiter forces to enable RTC ADC2 CTRL. (R/W)

**APB\_SARADC\_ADC\_ARB\_WIFI\_FORCE** ADC2 arbiter forces to enable PWDET/PKDET CTRL.  
(R/W)

**APB\_SARADC\_ADC\_ARB\_GRANT\_FORCE** ADC2 arbiter force grant. (R/W)

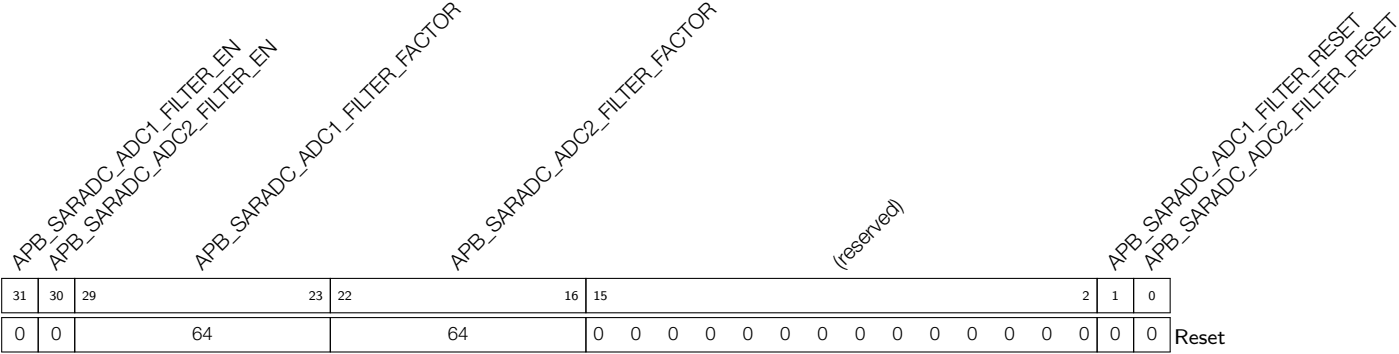
**APB\_SARADC\_ADC\_ARB\_APB\_PRIORITY** Set DIG ADC2 CTRL priority. (R/W)

**APB\_SARADC\_ADC\_ARB\_RTC\_PRIORITY** Set RTC ADC2 CTRL priority. (R/W)

**APB\_SARADC\_ADC\_ARB\_WIFI\_PRIORITY** Set PWDET/PKDET CTRL priority. (R/W)

**APB\_SARADC\_ADC\_ARB\_FIX\_PRIORITY** ADC2 arbiter uses fixed priority. (R/W)

Register 27.28: APB\_SARADC\_FILTER\_CTRL\_REG (0x003C)



**APB\_SARADC\_ADC2\_FILTER\_RESET** Reset ADC2 filter. (R/W)

**APB\_SARADC\_ADC1\_FILTER\_RESET** Reset ADC1 filter. (R/W)

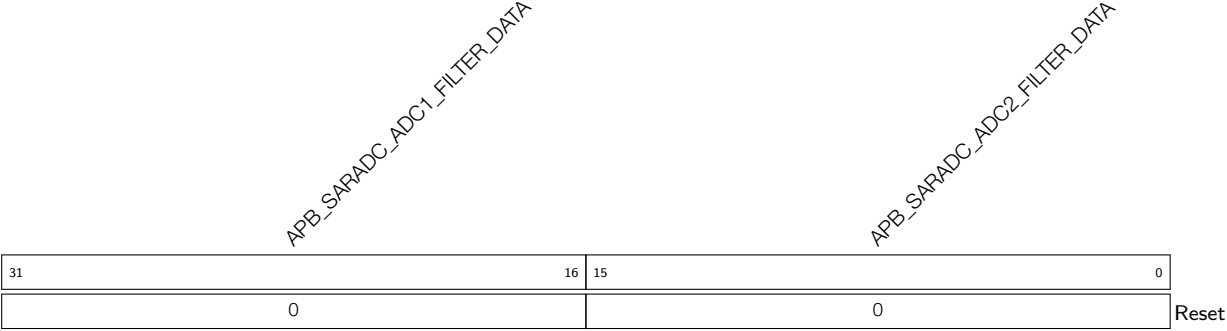
**APB\_SARADC\_ADC2\_FILTER\_FACTOR** Set filter factor for DIG ADC2 CRTL. (R/W)

**APB\_SARADC\_ADC1\_FILTER\_FACTOR** Set filter factor for DIG ADC1 CRTL. (R/W)

**APB\_SARADC\_ADC2\_FILTER\_EN** Enable DIG ADC2 CRTL filter. (R/W)

**APB\_SARADC\_ADC1\_FILTER\_EN** Enable DIG ADC1 CRTL filter. (R/W)

Register 27.29: APB\_SARADC\_FILTER\_STATUS\_REG (0x0040)



**APB\_SARADC\_ADC2\_FILTER\_DATA** ADC2 filter data. (RO)

**APB\_SARADC\_ADC1\_FILTER\_DATA** ADC1 filter data. (RO)

## 598

ESP32-S2 TRM (Preliminary V0.5)

[Submit Documentation Feedback](#)

**APB\_SARADC\_ADC1\_THRES\_EN** Enable ADC1 threshold monitor. (R/W)

**Register 27.31: APB\_SARADC\_INT\_ENA\_REG (0x0048)**

<div>APB_SARADC_ADC1_DONE_INT_ENA APB_SARADC_ADC2_DONE_INT_ENA APB_SARADC_ADC1_THRES_INT_ENA APB_SARADC_ADC2_THRES_INT_ENA  </div>																															
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

**APB\_SARADC\_ADC2\_THRES\_INT\_ENA** Enable bit of [APB\\_SARADC\\_ADC2\\_THRES\\_INT](#) interrupt.  
(R/W)

**APB\_SARADC\_ADC1\_THRES\_INT\_ENA** Enable bit of [APB\\_SARADC\\_ADC1\\_THRES\\_INT](#) interrupt.  
(R/W)

**APB\_SARADC\_ADC2\_DONE\_INT\_ENA** Enable bit of [APB\\_SARADC\\_ADC2\\_DONE\\_INT](#) interrupt.  
(R/W)

**APB\_SARADC\_ADC1\_DONE\_INT\_ENA** Enable bit of [APB\\_SARADC\\_ADC1\\_DONE\\_INT](#) interrupt.  
(R/W)

**Register 27.32: APB\_SARADC\_INT\_RAW\_REG (0x004C)**

<div style="position: relative; height: 100px;"><div style="position: absolute; top: 0; left: 0; transform: rotate(-45deg); font-size: small;">APB_SARADC_ADC1_DONE_INT_RAW APB_SARADC_ADC2_DONE_INT_RAW APB_SARADC_ADC1_THRES_INT_RAW APB_SARADC_ADC2_THRES_INT_RAW</div><div style="position: absolute; bottom: 0; right: 0; transform: rotate(45deg); font-size: small;">(reserved)</div></div>																																	
31	30	29	28	27																											0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**APB\_SARADC\_ADC2\_THRES\_INT\_RAW** Raw bit of [APB\\_SARADC\\_ADC2\\_THRES\\_INT](#) interrupt.  
(RO)

**APB\_SARADC\_ADC1\_THRES\_INT\_RAW** Raw bit of [APB\\_SARADC\\_ADC1\\_THRES\\_INT](#) interrupt.  
(RO)

**APB\_SARADC\_ADC2\_DONE\_INT\_RAW** Raw bit of [APB\\_SARADC\\_ADC2\\_DONE\\_INT](#) interrupt.  
(RO)

**APB\_SARADC\_ADC1\_DONE\_INT\_RAW** Raw bit of [APB\\_SARADC\\_ADC1\\_DONE\\_INT](#) interrupt.  
(RO)

Register 27.33: APB\_SARADC\_INT\_ST\_REG (0x0050)

 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|-----------------------------|----|----|----|----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|-----------------------------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|------------------------------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|------------------------------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|------------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
APB_SARADC_ADC1_DONE_INT_ST																															APB_SARADC_ADC2_DONE_INT_ST																															APB_SARADC_ADC1_THRES_INT_ST																															APB_SARADC_ADC2_THRES_INT_ST																															(reserved)																																																																																																				0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
31	30	29	28	27																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																

Reset

**APB\_SARADC\_ADC2\_THRES\_INT\_ST** Status of [APB\\_SARADC\\_ADC2\\_THRES\\_INT](#) interrupt. (RO)

**APB\_SARADC\_ADC1\_THRES\_INT\_ST** Status of [APB\\_SARADC\\_ADC1\\_THRES\\_INT](#) interrupt. (RO)

**APB\_SARADC\_ADC2\_DONE\_INT\_ST** Status of [APB\\_SARADC\\_ADC2\\_DONE\\_INT](#) interrupt. (RO)

**APB\_SARADC\_ADC1\_DONE\_INT\_ST** Status of [APB\\_SARADC\\_ADC1\\_DONE\\_INT](#) interrupt. (RO)

Register 27.34: APB\_SARADC\_INT\_CLR\_REG (0x0054)

APB_SARADC_ADC1_DONE_INT_CLR																															APB_SARADC_ADC2_DONE_INT_CLR																															APB_SARADC_ADC1_THRES_INT_CLR																															APB_SARADC_ADC2_THRES_INT_CLR																															(reserved)																																																																																																				0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
31	30	29	28	27																																																																																																																																																																																																																																	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

**APB\_SARADC\_ADC2\_THRES\_INT\_CLR** Clear bit of [APB\\_SARADC\\_ADC2\\_THRES\\_INT](#) interrupt. (WO)

**APB\_SARADC\_ADC1\_THRES\_INT\_CLR** Clear bit of [APB\\_SARADC\\_ADC1\\_THRES\\_INT](#) interrupt. (WO)

**APB\_SARADC\_ADC2\_DONE\_INT\_CLR** Clear bit of [APB\\_SARADC\\_ADC2\\_DONE\\_INT](#) interrupt. (WO)

**APB\_SARADC\_ADC1\_DONE\_INT\_CLR** Clear bit of [APB\\_SARADC\\_ADC1\\_DONE\\_INT](#) interrupt. (WO)



**Register 27.35: APB\_SARADC\_DMA\_CONF\_REG (0x0058)**

APB_SARADC_APB_ADC_TRANS																APB_SARADC_APB_ADC_RESET_FSM																(reserved)																APB_SARADC_APB_ADC_EOF_NUM															
31	30	29														16	15											0																																			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset															255																																

**APB\_SARADC\_APB\_ADC\_EOF\_NUM** Generate dma\_in\_suc\_eof when sample cnt = spi\_eof\_num.  
(R/W)

**APB\_SARADC\_APB\_ADC\_RESET\_FSM** Reset DIG ADC CTRL status. (R/W)

**APB\_SARADC\_APB\_ADC\_TRANS** Set this bit, DIG ADC CTRL uses SPI DMA. (R/W)

**Register 27.36: APB\_SARADC\_APB\_DAC\_CTRL\_REG (0x0060)**

(reserved)																APB_SARADC_APB_DAC_RST						APB_SARADC_APB_DAC_RESET_FIFO						APB_SARADC_APB_DAC_TRANS						APB_SARADC_APB_DAC_ALTER_MODE						APB_SARADC_DAC_TIMER_TARGET					
31																17	16	15	14	13	12	11											0												
0																0	0	0	1	0	100											Reset													

**APB\_SARADC\_DAC\_TIMER\_TARGET** Set DAC timer target. (R/W)

**APB\_SARADC\_DAC\_TIMER\_EN** Enable read dac data. (R/W)

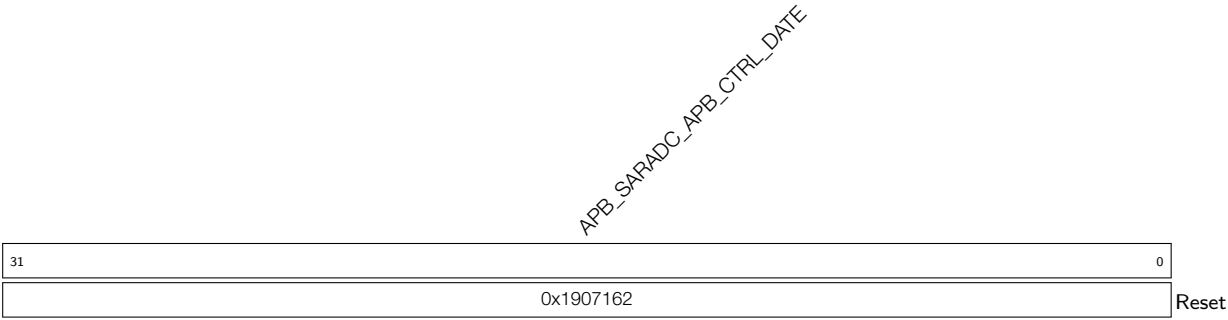
**APB\_SARADC\_APB\_DAC\_ALTER\_MODE** Enable DAC alter mode. (R/W)

**APB\_SARADC\_APB\_DAC\_TRANS** Enable DMA\_DAC. (R/W)

**APB\_SARADC\_DAC\_RESET\_FIFO** Reset DIG DAC FIFO. (R/W)

**APB\_SARADC\_APB\_DAC\_RST** Reset DIG DAC by software. (R/W)

Register 27.37: APB\_SARADC\_APB\_CTRL\_DATE\_REG (0x03FC)



APB\_SARADC\_APB\_CTRL\_DATE Version control register (R/W)

## 28. ULP Coprocessor

### 28.1 Overview

The ULP coprocessor is an ultra-low-power processor that remains powered on when the chip is in Deep-sleep (see Chapter 29 *Low-power Management*). Hence, users can store in RTC memory a program for the ULP coprocessor to access peripheral devices, internal sensors and RTC registers during Deep-sleep.

In power-sensitive scenarios, the main CPU goes to sleep mode to lower power consumption. Meanwhile, the coprocessor is woken up by ULP timer, and then monitors the external environment or interacts with the external circuit by controlling peripheral devices such as RTCIO, RTC I2C, SAR ADC, or temperature sensor (TSENS). The coprocessor wakes the main CPU up once a wakeup condition is reached.

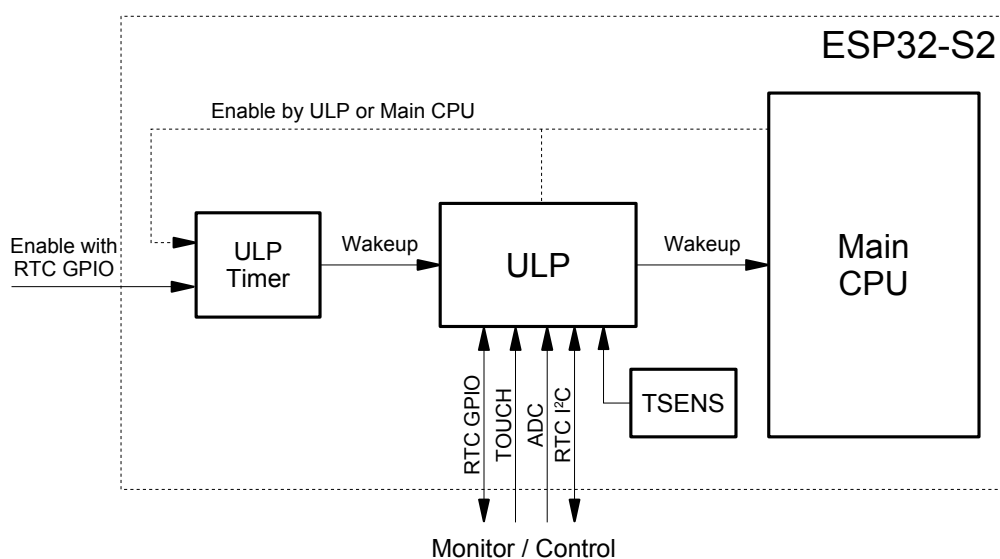


Figure 28-1. ULP Coprocessor Overview

ESP32-S2 has two ULP coprocessors, with one based on RISC-V instruction set architecture (ULP-RISC-V) and the other on finite state machine (ULP-FSM). Users can choose between the two coprocessors depending on their needs.

### 28.2 Features

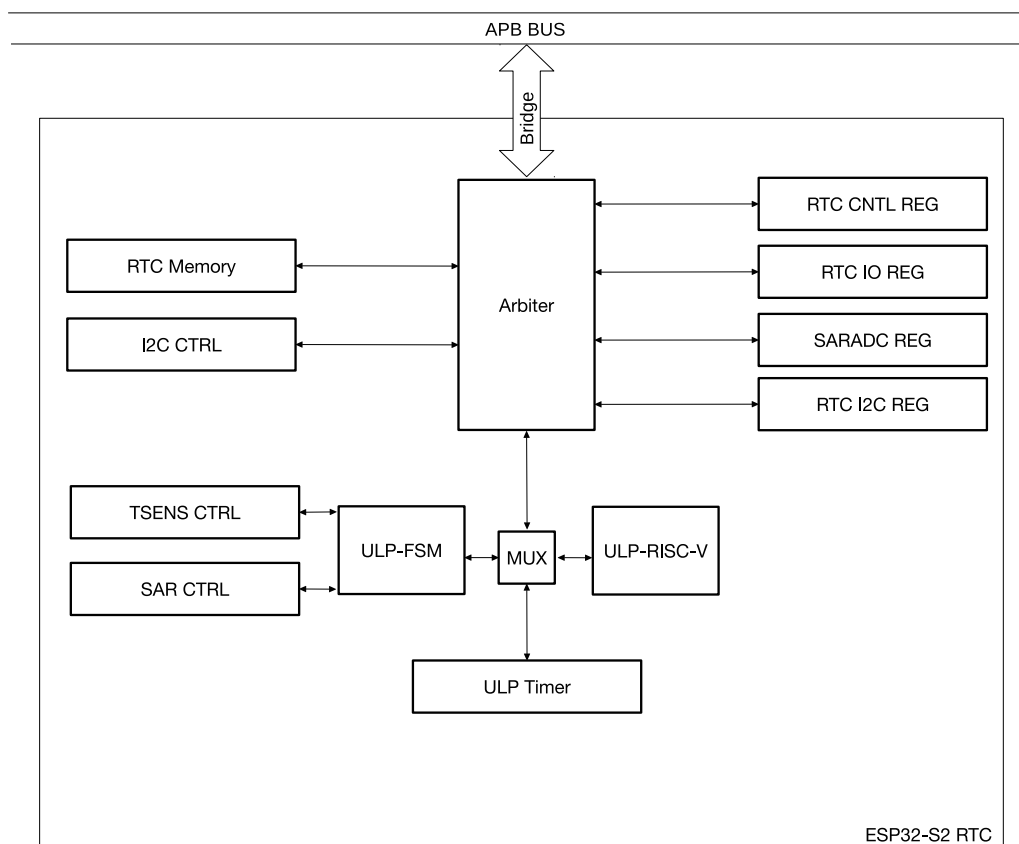
- Access up to 8 KB of SRAM RTC slow memory for instructions and data
- Clocked with 8 MHz RTC\_FAST\_CLK
- Support working in normal mode and in monitor mode
- Wake up the CPU or send an interrupt to the CPU
- Access peripherals, internal sensors and RTC registers

ULP-FSM and ULP-RISC-V can not be used simultaneously. Users can only choose one of them as the ULP coprocessor of ESP32-S2. The differences between the two coprocessors are shown in the table below.

Feature		Coprocessors	
		ULP-FSM	ULP-RISC-V
Memory (RTC Slow Memory)		8 KB	
Work Clock Frequency		8 MHz	
Wakeup Source		ULP Timer	
Work Mode	Normal Mode	Assist the main CPU to complete some tasks after the system is woken up.	
	Monitor Mode	Control sensors to do tasks such as monitoring environment, when the system is in sleep.	
Control Low-Power Peripherals		ADC0/ADC1	
		DAC0/DAC1	
		RTC I2C	
		RTC GPIO	
		Touch Sensor	
		Temperature Sensor	
Architecture		Programmable FSM	RISC-V
Development		Special instruction set	Standard C compiler

**Table 149: Comparison of the Two Coprocessors**

ULP coprocessor can access the modules in RTC domain via RTC registers. In many cases the ULP coprocessor can be a good supplement to, or replacement of, the main CPU, especially for power-sensitive applications. Figure 28-2 shows the overall layout of ESP32-S2 coprocessor.

**Figure 28-2. ULP Coprocessor Diagram**

## 28.3 Programming Workflow

The ULP-RISC-V is intended for programming using C language. The program in C is then compiled to [RV32IMC](#) standard instruction code. The ULP-FSM is using custom instructions normally not supported by high-level programming language. Users develop their programs using ULP-FSM instructions (see Section [28.5.2](#)).

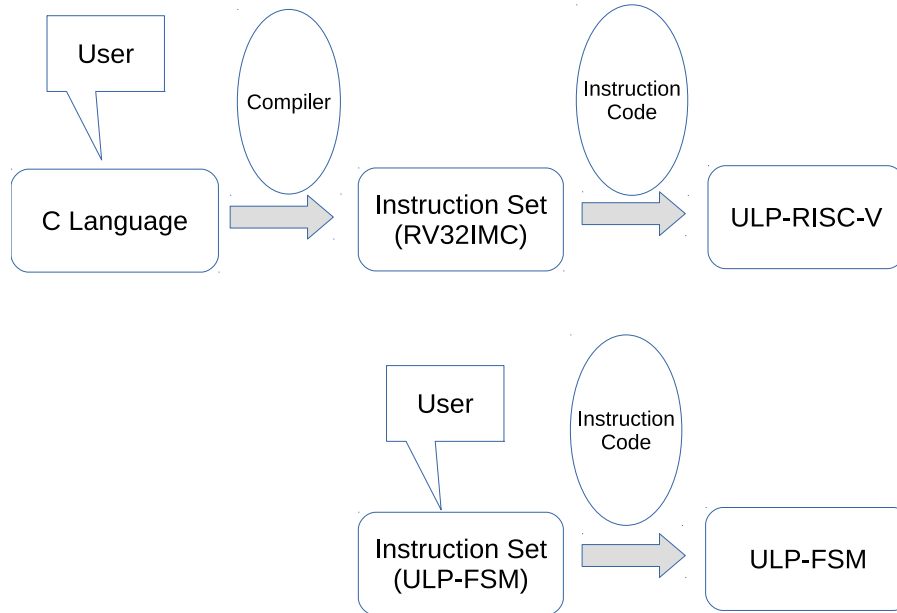


Figure 28-3. Programing Workflow

## 28.4 ULP Coprocessor Workflow

ULP coprocessor is designed to operate independently of the CPU, while the CPU is either in sleep or running.

In a typical power-saving scenario, the chip goes to Deep-sleep mode to lower power consumption. Before setting the chip to sleep mode, users should complete the following operations.

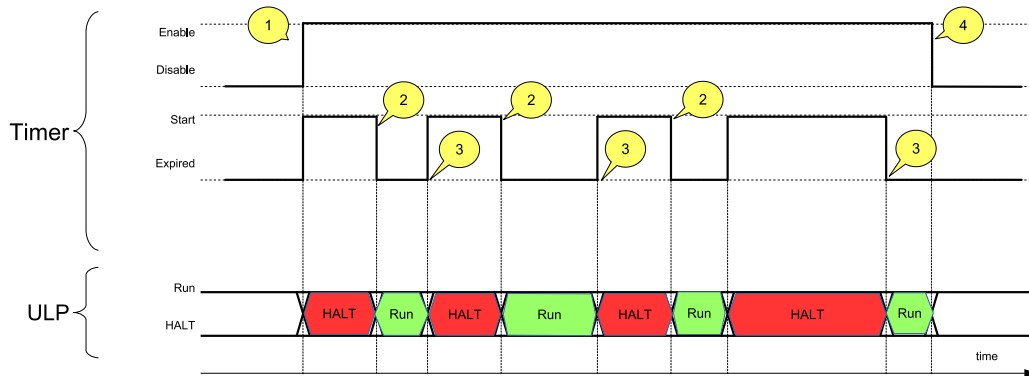
1. Flash the program to be executed by ULP coprocessor into RTC slow memory.
2. Select the working ULP coprocessor by configuring the register [RTC\\_CNTL\\_COCPU\\_SEL](#).
  - 0: select ULP-RISC-V
  - 1: select ULP-FSM
3. Set sleep cycles for the timer by configuring [RTC\\_CNTL\\_ULP\\_CP\\_TIMER\\_1\\_REG](#).
4. Enable the timer by software or by RTC GPIO;
  - By software: set the register [RTC\\_CNTL\\_ULP\\_CP\\_SLP\\_TIMER\\_EN](#).
  - By RTC GPIO: set the register [RTC\\_CNTL\\_ULP\\_CP\\_GPIO\\_WAKEUP\\_ENA](#).
5. Set the system into sleep mode.

When the system is in Deep-sleep mode:

1. The timer periodically sets the low-power controller (see Chapter [29 Low-power Management](#)) to Monitor mode and then wakes up the coprocessor.

2. Coprocessor executes some necessary operations, such as monitoring external environment via low-power sensors.
3. After the operations are finished, the system goes back to Deep-sleep mode.
4. ULP coprocessor goes back to halt mode and waits for next wakeup.

In monitor mode, ULP coprocessor is woken up and goes to halt as shown in Figure 28-4.



**Figure 28-4. Sample of a ULP Operation Sequence**

1. Enable the timer and the timer starts counting.
2. The timer expires and wakes up the ULP coprocessor. ULP coprocessor starts running and executes the program flashed in RTC slow memory.
3. ULP coprocessor goes to halt and the timer starts counting again.
  - Put ULP-RISC-V into HALT: set the register `RTC_CNTL_COCPU_DONE`,
  - Put ULP-FSM into HALT: execute HALT instruction.
4. Disable the timer by ULP program or by software. The system exits from monitor mode.
  - Disabled by software: clear the register `RTC_CNTL_ULP_CP_SLP_TIMER_EN`.
  - Disabled by RTC GPIO: clear the register `RTC_CNTL_ULP_CP_GPIO_WAKEUP_ENA`, and set the register `RTC_CNTL_ULP_CP_GPIO_WAKEUP_CLR`.

**Note:**

- If the timer is enabled by software (RTC GPIO), it should be disabled by software (RTC GPIO).
- Before setting ULP-RISC-V to HALT, users should configure the register `RTC_CNTL_COCPU_DONE` first, therefore, it is recommended to end the flashed program with the following pattern:
  - Set the register `RTC_CNTL_COCPU_DONE` to end the operation of ULP-RISC-V and put it into halt;
  - Set the register `RTC_CNTL_COCPU_SHUT_RESET_EN` to reset ULP-RISC-V.

Enough time is reserved for the ULP-RISC-V to complete the operations above before it goes to halt.

The relationship between the signals and registers is shown in Figure 28-5.

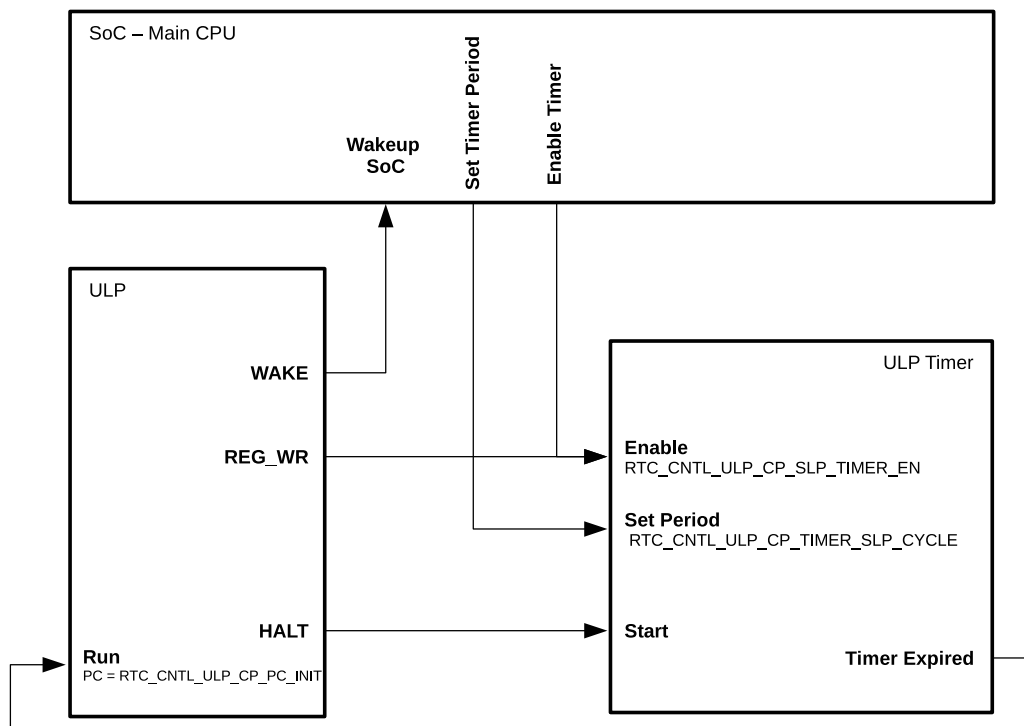


Figure 28-5. Control of ULP Program Execution

## 28.5 ULP-FSM

### 28.5.1 Features

ULP-FSM is a programmable finite state machine that can work while the main CPU is in Deep-sleep. ULP-FSM supports instructions for complex logic and arithmetic operations, and also provides dedicated instructions for RTC controllers or peripherals. ULP-FSM can access up to 8 KB of SRAM RTC slow memory (accessible by the CPU) for instructions and data. Hence, such memory is usually used to store instructions and share data between the ULP coprocessor and the CPU. ULP-FSM can be stopped by running HALT instruction.

ULP-FSM has the following features.

- Provide four 16-bit general-purpose registers (R0, R1, R2, and R3) for manipulating data and accessing memory.
- Provide one 8-bit stage count register (Stage\_cnt) which can be manipulated by ALU and used in JUMP instructions.
- Support built-in instructions specially for direct control of low-power peripherals, such as SAR ADC and temperature sensor.

### 28.5.2 Instruction Set

ULP-FSM supports the following instructions.

- ALU: perform arithmetic and logic operations
- LD, ST, REG\_RD and REG\_WR: load and store data
- JUMP: jump to a certain address
- WAIT/HALT: manage program execution

- WAKE: wake up CPU or communicate with the CPU
- TSENS and ADC: take measurements

The format of ULP-FSM instructions is shown in Figure 28-6.

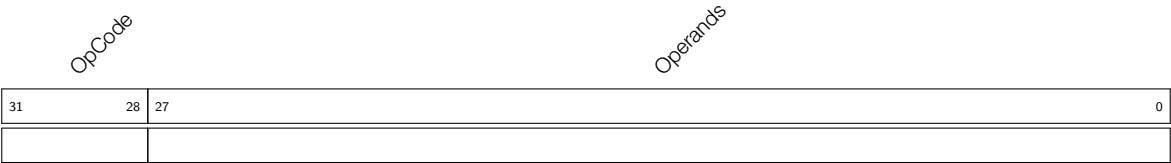


Figure 28-6. ULP-FSM Instruction Format

An instruction, which has one OpCode, can perform various operations, depending on the setting of Operands bits. A good example is the ALU instruction, which is able to perform 10 arithmetic and logic operations; or the JUMP instruction, which may be conditional or unconditional, absolute or relative.

Each instruction has a fixed width of 32 bits. A series of instructions can make a program be executed by the coprocessor. The execution flow inside the program uses 32-bit addressing. The program is stored in a dedicated region called Slow Memory, which is visible to the main CPU as one that has an address range of 0x5000\_0000 to 0x5000\_1FFF (8 KB).

28.5.2.1 ALU - Perform Arithmetic and Logic Operations

ALU (Arithmetic and Logic Unit) performs arithmetic and logic operations on values stored in ULP coprocessor registers, and on immediate values stored in the instruction itself. The following operations are supported.

- Arithmetic: ADD and SUB
- Logic: AND and OR
- Bit shifting: LSH and RSH
- Moving data to register: MOVE
- PC register operations - STAGE\_RST, STAGE\_INC, and STAGE\_DEC

The ALU instruction, which has one OpCode (7), can perform various arithmetic and logic operations, depending on the setting of the instruction bits [27:21].

Operations Among Registers

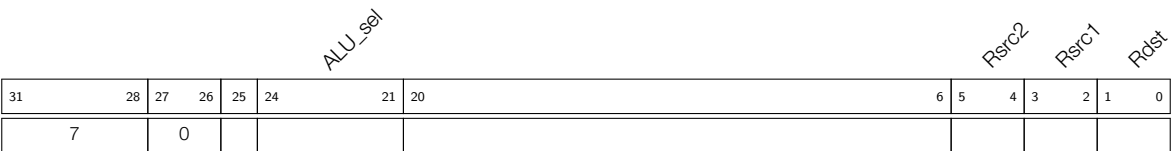


Figure 28-7. Instruction Type — ALU for Operations Among Registers

When bits [27:26] of the instruction in Figure 28-7 are set to 0, ALU performs operations on the data stored in ULP-FSM registers R[0-3]. The types of operations depend on the setting of the instruction bits ALU\_sel [24:21] presented in Table 150.



**Operand**    **Description** - see Figure 28-7

<i>Rdst</i>	Register R[0-3], destination
<i>Rsrc1</i>	Register R[0-3], source
<i>Rsrc2</i>	Register R[0-3], source
<i>ALU_sel</i>	ALU Operation

ALU_sel	Instruction	Operation	Description
0	ADD	$Rdst = Rsrc1 + Rsrc2$	Add to register
1	SUB	$Rdst = Rsrc1 - Rsrc2$	Subtract from register
2	AND	$Rdst = Rsrc1 \& Rsrc2$	Logical AND of two operands
3	OR	$Rdst = Rsrc1   Rsrc2$	Logical OR of two operands
4	MOVE	$Rdst = Rsrc1$	Move to register
5	LSH	$Rdst = Rsrc1 \ll Rsrc2$	Logical shift left
6	RSH	$Rdst = Rsrc1 \gg Rsrc2$	Logical shift right

Table 150: ALU Operations Among Registers

**Note:**

- ADD or SUB operations can be used to set or clear the overflow flag in ALU.
- All ALU operations can be used to set or clear the zero flag in ALU.

**Operations with Immediate Value**

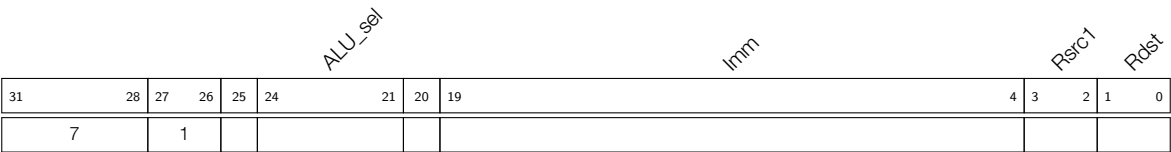


Figure 28-8. Instruction Type — ALU for Operations with Immediate Value

When bits [27:26] of the instruction in Figure 28-8 are set to 1, ALU performs operations using register R[0-3] and the immediate value stored in instruction bits [19:4]. The types of operations depend on the setting of the instruction bits ALU\_sel[24:21] presented in Table 151.

**Operand**    **Description** - see Figure 28-8

<i>Rdst</i>	Register R[0-3], destination
<i>Rsrc1</i>	Register R[0-3], source
<i>Imm</i>	16-bit signed immediate value
<i>ALU_sel</i>	ALU Operation

ALU_sel	Instruction	Operation	Description
0	ADD	$Rdst = Rsrc1 + Imm$	Add to register
1	SUB	$Rdst = Rsrc1 - Imm$	Subtract from register
2	AND	$Rdst = Rsrc1 \& Imm$	Logical AND of two operands
3	OR	$Rdst = Rsrc1 \mid Imm$	Logical OR of two operands
4	MOVE	$Rdst = Imm$	Move to register
5	LSH	$Rdst = Rsrc1 \ll Imm$	Logical shift left
6	RSH	$Rdst = Rsrc1 \gg Imm$	Logical shift right

Table 151: ALU Operations with Immediate Value

**Note:**

- ADD or SUB operations can be used to set or clear the overflow flag in ALU.
- All ALU operations can be used to set or clear the zero flag in ALU.

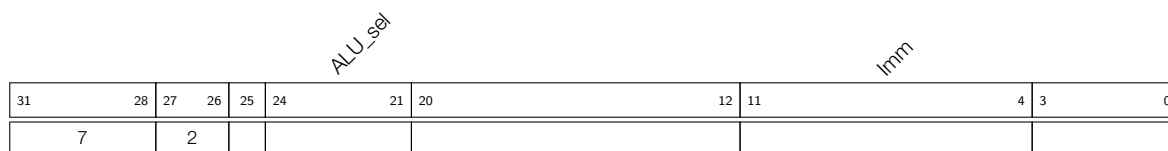
**Operations with Stage Count Register**

Figure 28-9. Instruction Type — ALU for Operations with Stage Count Register

ALU is also able to increment or decrement by a given value, or reset the 8-bit register Stage\_cnt. To do so, bits [27:26] of instruction in Figure 28-9 should be set to 2. The type of operation depends on the setting of the instruction bits ALU\_sel[24:21] presented in Table 28-9. The Stage\_cnt is a separate register and is not a part of the instruction in Figure 28-9.

**Operand Description** - see Figure 28-9

*Imm* 8-bit signed immediate value

*ALU\_sel* ALU Operation

*Stage\_cnt* Stage count register, a 8-bit separate register used to store variables, such as loop index

ALU_sel	Instruction	Operation	Description
0	STAGE_INC	$Stage\_cnt = Stage\_cnt + Imm$	Increment stage count register
1	STAGE_DEC	$Stage\_cnt = Stage\_cnt - Imm$	Decrement stage count register
2	STAGE_RST	$Stage\_cnt = 0$	Reset stage count register

Table 152: ALU Operations with Stage Count Register

**Note:** This instruction is mainly used with JUMPS instruction based on the stage count register to form a stage count for-loop. For the usage, please refer to the following pseudocode:

```
STAGE_RST // clear stage count register
STAGE_INC // stage count register ++
{...} // loop body, containing n instructions
JUMPS (step = n, cond = 0, threshold = m) // If the value of stage count register is less than m, then jump to
STAGE_INC, otherwise jump out of the loop. By such way, a cumulative for-loop with threshold m is implemented.
```

28.5.2.2 ST – Store Data in Memory

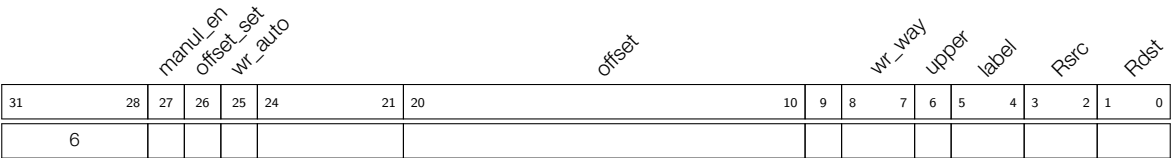


Figure 28-10. Instruction Type - ST

Operand	Description - see Figure 28-10
Rdst	Register R[0-3], address of the destination, expressed in 32-bit words
Rsrc	Register R[0-3], 16-bit value to store
label	Data label, 2-bit user defined unsigned value
upper	0: write the low half-word; 1: write the high half-word
wr_way	0: write the full-word; 1: with the label; 3: without the label
offset	11-bit signed value, expressed in 32-bit words
wr_auto	Enable automatic storage mode
offset_set	Offset enable bit. 0: Do not configure the offset for automatic storage mode. 1: Configure the offset for automatic storage mode.
manul_en	Enable manual storage mode

Automatic Storage Mode

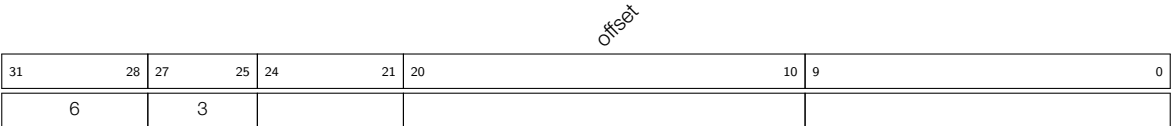


Figure 28-11. Instruction Type - Offset in Automatic Storage Mode (ST-OFFSET)

Operand	Description - see Figure 28-11
offset	Initial address offset, 11-bit signed value, expressed in 32-bit words

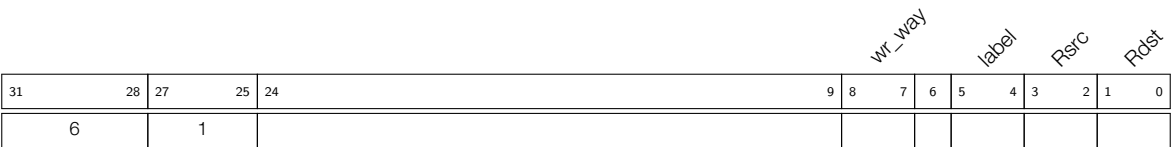


Figure 28-12. Instruction Type - Data Storage in Automatic Storage Mode (ST-AUTO-DATA)

<b>Operand</b>	<b>Description</b> - See Figure 28-12
<i>Rdst</i>	Register R[0-3], address of the destination, expressed in 32-bit words
<i>Rsrc</i>	Register R[0-3], 16-bit value to store
<i>label</i>	Data label, 2-bit user defined unsigned value
<i>wr_way</i>	0: write the fullword; 1: with the label; 3: without the label

Description

This mode is used to access continuous addresses. Before using this mode for the first time, please configure the initial address using ST-OFFSET instruction. Executing the instruction ST-AUTO-DATA will store the 16-bit data in *Rsrc* into the memory address *Rdst* + *Offset*, see Table 153. Write\_cnt here indicates the times of the instruction ST-AUTO-DATA executed.

wr_way	write_cnt	Store Data	Operation
0	*	Mem [Rdst + Offset]{31:0} = {PC[10:0], 5'b0, Rsrc[15:0]}	Write full-word, including the pointer and the data
1	odd	Mem [Rdst + Offset]{15:0} = {Label[1:0],Rsrc[13:0]}	Store the data with label in the low half-word
1	even	Mem [Rdst + Offset]{31:16} = {Label[1:0],Rsrc[13:0]}	Store the data with label in the high half-word
3	odd	Mem [Rdst + Offset]{15:0} = Rsrc[15:0]	Store the data without label in the low half-word
3	even	Mem [Rdst + Offset]{31:16} = Rsrc[15:0]	Store the data without label in the high half-word

Table 153: Data Storage Type - Automatic Storage Mode

Note:

- When full-word is written, the offset will be automatically incremented by 1 after each ST-AUTO-DATA execution.
- When half-word is written (low half-word first), the offset will be automatically incremented by 1 after twice ST-AUTO-DATA execution.
- This instruction can only access 32-bit memory words.
- The "Mem" written is the RTC\_SLOW\_MEM memory. Address 0, as seen by the ULP coprocessor, corresponds to address 0x50000000, as seen by the main CPU.

Manual Storage Mode

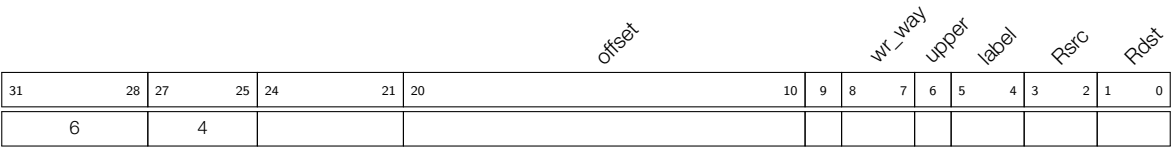


Figure 28-13. Instruction Type - Data Storage in Manual Storage Mode

**Operand Description** - See Figure 28-13

<i>Rdst</i>	Register R[0-3], address of the destination, expressed in 32-bit words
<i>Rsrc</i>	Register R[0-3], 16-bit value to store
<i>label</i>	Data label, 2-bit user defined unsigned value
<i>upper</i>	0: Write the low half-word; 1: write the high half-word
<i>wr_way</i>	0: Write the full-word; 1: with the label; 3: without the label
<i>offset</i>	11-bit signed value, expressed in 32-bit words

**Description**

Manual storage mode is mainly used for storing data into discontinuous addresses. Each instruction needs a storage address and offset. The detailed storage methods are shown in Table 154.

<b>wr_way</b>	<b>upper</b>	<b>Data</b>	<b>Operation</b>
0	*	Mem [Rdst + Offset]{31:0} = {PC[10:0], 5'b0, Rsrc[15:0]}	Write full-word, including the pointer and the data
1	0	Mem [Rdst + Offset]{15:0} = {Label[1:0], Rsrc[13:0]}	Store the data with label in the low half-word
1	1	Mem [Rdst + Offset]{31:16} = {Label[1:0], Rsrc[13:0]}	Store the data with label in the high half-word
3	0	Mem [Rdst + Offset]{15:0} = Rsrc[15:0]	Store the data without label in the low half-word
3	1	Mem [Rdst + Offset]{31:16} = Rsrc[15:0]	Store the data without label in the high half-word

**Table 154: Data Storage - Manual Storage Mode****28.5.2.3 LD – Load Data from Memory****Figure 28-14. Instruction Type - LD****Operand Description** - see Figure 28-14

<i>Rdst</i>	Register R[0-3], destination
<i>Rsrc</i>	Register R[0-3], address of destination memory, expressed in 32-bit words
<i>Offset</i>	11-bit signed value, expressed in 32-bit words
<i>rd_upper</i>	Choose which half-word to read: 0 - read the high half-word 1 - read the low half-word

**Description**

This instruction loads the low or high 16-bit half-word, depending on *rd\_upper*, from memory with address *Rsrc* + *offset* into the destination register *Rdst*:

$$Rdst[15:0] = Mem[Rsrc + Offset]$$

**Note:**

- This instruction can only access 32-bit memory words.
- The “Mem” loaded is the RTC\_SLOW\_MEM memory. Address 0, as seen by the ULP coprocessor, corresponds to address 0x50000000, as seen by the main CPU.

**28.5.2.4 JUMP – Jump to an Absolute Address**

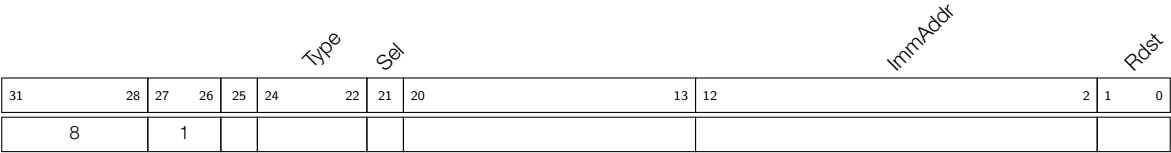


Figure 28-15. Instruction Type- JUMP

Operand	Description - see Figure 28-15
<i>Rdst</i>	Register R[0-3], containing address to jump to (expressed in 32-bit words)
<i>ImmAddr</i>	11-bit address, expressed in 32-bit words
<i>Sel</i>	Select the address to jump to: 0 - jump to the address stored in <i>ImmAddr</i> 1 - jump to the address stored in <i>Rdst</i>
<i>Type</i>	Jump type: 0 - make an unconditional jump 1 - jump only if the last ALU operation has set zero flag 2 - jump only if the last ALU operation has set overflow flag

**Note:**

All jump addresses are expressed in 32-bit words.

**Description**

The instruction executes a jump to a specified address. The jump can be either unconditional or based on the ALU flag.

**28.5.2.5 JUMPR – Jump to a Relative Offset (Conditional upon R0)**

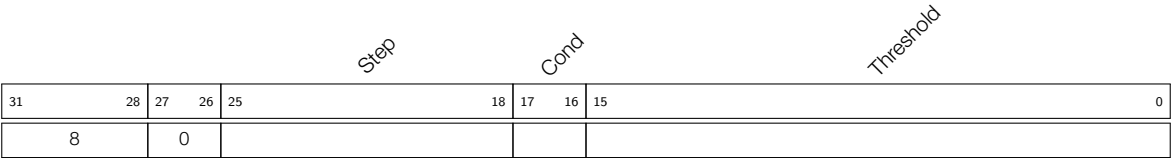


Figure 28-16. Instruction Type - JUMPR

**Operand**    **Description** - see Figure 28-16*Threshold*    Threshold value for condition (see *Cond* below) to jump*Cond*    Condition to jump:0 - jump if  $R0 < Threshold$ 1 - jump if  $R0 > Threshold$ 2 - jump if  $R0 = Threshold$ *Step*    Relative shift from current position, expressed in 32-bit words:if  $Step[7] = 0$ , then  $PC = PC + Step[6:0]$ if  $Step[7] = 1$ , then  $PC = PC - Step[6:0]$ **Note:**

All jump addresses are expressed in 32-bit words.

**Description**

The instruction executes a jump to a relative address, if the above-mentioned condition is true. The condition is the result of comparing the R0 register value and the *Threshold* value.

### 28.5.2.6 JUMPS – Jump to a Relative Address (Conditional upon Stage Count Register)

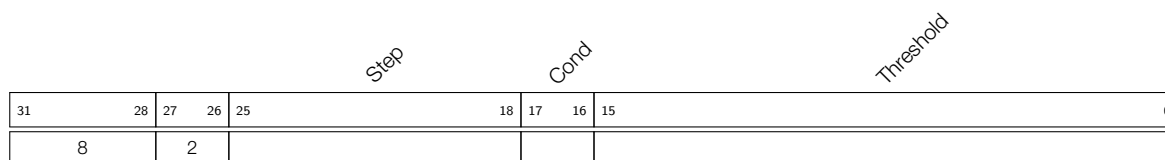


Figure 28-17. Instruction Type - JUMPS

**Operand**    **Description** - see Figure 28-17*Threshold*    Threshold value for condition (see *Cond* below) to jump*Cond*    Condition to jump:1X - jump if  $Stage\_cnt \leq Threshold$ 00 - jump if  $Stage\_cnt < Threshold$ 01 - jump if  $Stage\_cnt \geq Threshold$ *Step*    Relative shift from current position, expressed in 32-bit words:if  $Step[7] = 0$ , then  $PC = PC + Step[6:0]$ if  $Step[7] = 1$ , then  $PC = PC - Step[6:0]$ **Note:**

- For more information about the stage count register, please refer to Section 28.5.2.1.
- All jump addresses are expressed in 32-bit words.

**Description**

The instruction executes a jump to a relative address if the above-mentioned condition is true. The condition itself is the result of comparing the value of *Stage\_cnt* (stage count register) and the *Threshold* value.

### 28.5.2.7 HALT – End the Program

31	28	27																		0
11																				

Figure 28-18. Instruction Type- HALT

#### Description

The instruction ends the operation of the ULP-FSM and puts it into power-down mode.

#### Note:

After executing this instruction, the ULP coprocessor wakeup timer gets started.

### 28.5.2.8 WAKE – Wake up the Chip

31	28	27	26	25																1	0
9		0																			1'b

Figure 28-19. Instruction Type - WAKE

#### Description

This instruction sends an interrupt from the ULP-FSM to the RTC controller.

- If the chip is in Deep-sleep mode, and the ULP wakeup timer is enabled, the above-mentioned interrupt will wake up the chip.
- If the chip is not in Deep-sleep mode, and the ULP interrupt bit (RTC\_CNTL\_ULP\_CP\_INT\_ENA) is set in register RTC\_CNTL\_INT\_ENA\_REG, an RTC interrupt will be triggered.

### 28.5.2.9 WAIT – Wait for a Number of Cycles

															Cycles																					
31	28	27														16	15														0					
4																																				

Cycles

Figure 28-20. Instruction Type - WAIT

**Operand**    **Description** - see Figure 28-20

*Cycles*        The number of cycles to wait

#### Description

The instruction will delay the ULP-FSM for a given number of cycles.



### 28.5.2.10 TSENS – Take Measurement with Temperature Sensor

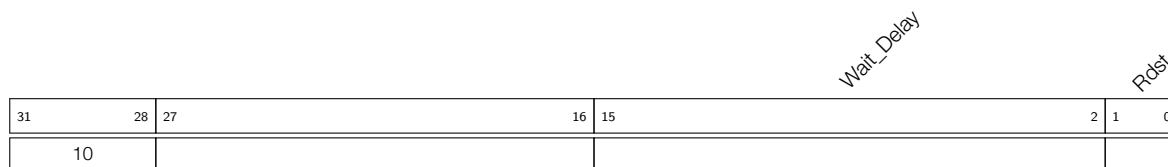


Figure 28-21. Instruction Type - TSENS

**Operand**     **Description** - see Figure 28-21

*Rdst*            Destination Register R[0-3], results will be stored in this register.

*Wait\_Delay*    Number of cycles used to perform the measurement.

#### Description

Increasing the measurement cycles *Wait\_Delay* helps improve the accuracy and optimize the result. The instruction performs measurement via temperature sensor and stores the result into a general purpose register.

### 28.5.2.11 ADC – Take Measurement with ADC

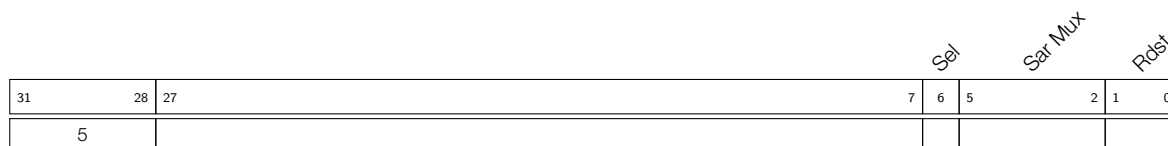


Figure 28-22. Instruction Type - ADC

**Operand**     **Description** - see Figure 28-22

*Rdst*            Destination Register R[0-3], results will be stored in this register.

*Sar\_Mux*       Enable SAR ADC pad [Sar\_Mux - 1], see Table 155.

*Sel*             Select ADC. 0: select SAR ADC1; 1: select SAR ADC2, see Table 155.

Table 155: Input Signals Measured Using the ADC Instruction

Pad / Signal / GPIO	<i>Sar_Mux</i>	ADC Selection <i>Sel</i>
GPIO1	1	<i>Sel</i> = 0, select SAR ADC1
GPIO2	2	
GPIO3	3	
GPIO4	4	
GPIO5	5	
GPIO6	6	
GPIO7	7	
GPIO8	8	
GPIO9	9	
GPIO10	10	
GPIO11	1	<i>Sel</i> = 1, select SAR ADC2
GPIO12	2	

Pad / Signal / GPIO	<i>Sar_Mux</i>	ADC Selection <i>Sel</i>
GPIO13	3	<i>Sel</i> = 1, select SAR ADC2
GPIO14	4	
XTAL_32k_P	5	
XTAL_32k_N	6	
DAC_1	7	
DAC_2	8	
GPIO19	9	
GPIO20	10	

### 28.5.2.12 REG\_RD – Read from Peripheral Register

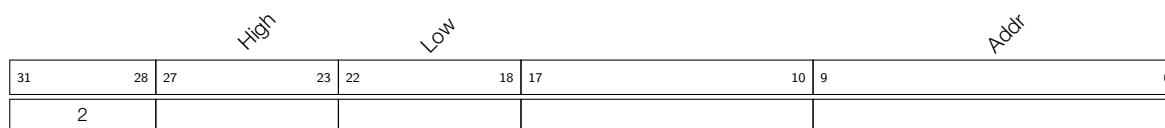


Figure 28-23. Instruction Type - REG\_RD

**Operand**    **Description** - see Figure 28-23

*Addr*        Peripheral register address, in 32-bit words

*Low*        Register start bit number

*High*       Register end bit number

#### Description

The instruction reads up to 16 bits from a peripheral register into a general-purpose register:

$$R0 = \text{REG}[\text{Addr}][\text{High}:\text{Low}]$$

In case of more than 16 bits being requested, i.e.  $\text{High} - \text{Low} + 1 > 16$ , then the instruction will return  $[\text{Low}+15:\text{Low}]$ .

#### Note:

- This instruction can access registers in RTC\_CNTL, RTC\_IO, SENS, and RTC\_I2C peripherals. Address of the register, as seen from the ULP coprocessor, can be calculated from the address of the same register on PeriBUS1 (*addr\_peribus1*), as follows:

$$\text{addr\_ulp} = (\text{addr\_peribus1} - \text{DR\_REG\_RTCCNTL\_BASE}) / 4$$

- The *addr\_ulp* is expressed in 32-bit words (not in bytes), and value 0 maps onto the DR\_REG\_RTCCNTL\_BASE (as seen from the main CPU). Thus, 10 bits of address cover a 4096-byte range of peripheral register space, including regions DR\_REG\_RTCCNTL\_BASE, DR\_REG\_RTCIO\_BASE, DR\_REG\_SENS\_BASE, and DR\_REG\_RTC\_I2C\_BASE.

### 28.5.2.13 REG\_WR – Write to Peripheral Register

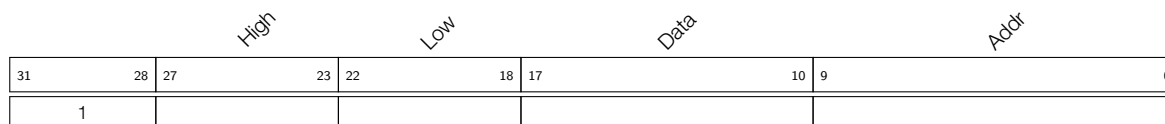


Figure 28-24. Instruction Type - REG\_WR

**Operand**    **Description** - see Figure 28-24

*Addr*        Register address, expressed in 32-bit words

*Data*        Value to write, 8 bits

*Low*         Register start bit number

*High*        Register end bit number

#### Description

This instruction writes up to 8 bits from an immediate data value into a peripheral register.

$$\text{REG}[\text{Addr}][\text{High}:\text{Low}] = \text{Data}$$

If more than 8 bits are requested, i.e.  $\text{High} - \text{Low} + 1 > 8$ , then the instruction will pad with zeros the bits above the eighth bit.

#### Note:

See notes regarding *addr\_ulp* in Section 28.5.2.12.

## 28.6 ULP-RISC-V

### 28.6.1 Features

- Support [RV32IMC](#) instruction set
- Thirty-two 32-bit general-purpose registers
- 32-bit multiplier and divider
- Support for interrupts

### 28.6.2 Multiplier and Divider

ULP-RISC-V has an independent multiplication and division unit. The efficiency of multiplication and division instructions is shown in the following table.

**Table 156: Instruction Efficiency**

Operation	Instruction	Execution Cycle	Instruction Description
Multiply	MUL	34	Multiply two 32-bit integers and return the lower 32-bit of the result
	MULH	66	Multiply two 32-bit signed integers and return the higher 32-bit of the result
	MULHU	66	Multiply two 32-bit unsigned integers and return the higher 32-bit of the result
	MULHSU	66	Multiply a 32-bit signed integer with a unsigned integer and return the higher 32-bit of the result
Divide	DIV	34	Divide a 32-bit integer by a 32-bit integer and return the quotient
	DIVU	34	Divide a 32-bit unsigned integer by a 32-bit unsigned integer and return the quotient
	REM	34	Divide a 32-bit signed integer by a 32-bit signed integer and return the remainder
	REMU	34	Divide a 32-bit unsigned integer by a 32-bit unsigned integer and return the remainder

### 28.6.3 ULP-RISC-V Interrupts

The interrupts from some sensors, software, and RTC I2C can be routed to ULP-RISC-V. To enable the interrupts, please set the register [SENS\\_SAR\\_COCPU\\_INT\\_ENA\\_REG](#), see Table 157.

Enable bit	Interrupt	Description
0	TOUCH_DONE_INT	Triggered when the touch sensor completes the scan of a channel
1	TOUCH_INACTIVE_INT	Triggered when the touch pad is released
2	TOUCH_ACTIVE_INT	Triggered when the touch pad is touched
3	SARADC1_DONE_INT	Triggered when SAR ADC1 completes the conversion one time
4	SARADC2_DONE_INT	Triggered when SAR ADC2 completes the conversion one time
5	TSENS_DONE_INT	Triggered when the temperature sensor completes the dump of its data
6	RISCV_START_INT	Triggered when ULP-RISC-V powers on and starts working
7	SW_INT	Triggered by software
8	SWD_INT	Triggered by timeout of Super Watchdog (SWD)

**Table 157: ULP-RISC-V Interrupt List**

**Note:**

- Besides the above-mentioned interrupts, ULP-RISC-V can also handle the interrupt from RTC\_IO by simply configuring RTC\_IO as input mode. Users can configure [RTCIO\\_GPIO\\_PIN<sub>n</sub>\\_INT\\_TYPE](#) to select the interrupt trigger modes, but only level trigger modes are available. For more details about RTC\_IO configuration, see Chapter IO MUX and GPIO Matrix.
- The interrupt from RTC\_IO can be cleared by releasing RTC\_IO and its source can be read from the register [RTCIO\\_RTC\\_GPIO\\_STATUS\\_REG](#).
- The SW\_INT interrupt is generated by configuring the register [RTC\\_CNTL\\_COCPU\\_SW\\_INT\\_TRIGGER](#).
- For the information about RTC I2C interrupts, please refer to Section 28.7.4.

## 28.7 RTC I2C Controller

ULP coprocessor can use RTC I2C controller to read from or write to the external I2C slave devices.

### 28.7.1 Connecting RTC I2C Signals

SDA and SCL signals can be mapped onto two out of the four GPIO pins, which are identified in Table 24 in Chapter 5 *IO MUX and GPIO Matrix*, using the register `RTCIO_SAR_I2C_IO_REG`.

### 28.7.2 Configuring RTC I2C

Before the ULP coprocessor can use the I2C instruction, certain parameters of the RTC I2C need to be configured. Configuration is performed by writing certain timing parameters into the RTC I2C registers. This can be done by the program running on the main CPU, or by the ULP coprocessor itself.

1. Set the low and high SCL half-periods by configuring `RTC_I2C_SCL_LOW_PERIOD_REG` and `RTC_I2C_SCL_HIGH_PERIOD_REG` in `RTC_FAST_CLK` cycles (e.g. `RTC_I2C_SCL_LOW_PERIOD_REG` = 40, `RTC_I2C_SCL_HIGH_PERIOD_REG` = 40 for 100 kHz frequency).
2. Set the number of cycles between the SDA switch and the falling edge of SCL by using `RTC_I2C_SDA_DUTY_REG` in `RTC_FAST_CLK` (e.g. `RTC_I2C_SDA_DUTY_REG` = 16).
3. Set the waiting time after the START condition by using `RTC_I2C_SCL_START_PERIOD_REG` (e.g. `RTC_I2C_SCL_START_PERIOD` = 30).
4. Set the waiting time before the END condition by using `RTC_I2C_SCL_STOP_PERIOD_REG` (e.g. `RTC_I2C_SCL_STOP_PERIOD` = 44).
5. Set the transaction timeout by using `RTC_I2C_TIME_OUT_REG` (e.g. `RTC_I2C_TIME_OUT_REG` = 200).
6. Configure the RTC I2C controller into master mode by setting the `RTC_I2C_MS_MODE` bit in `RTC_I2C_CTRL_REG`.
7. Write the address(es) of external slave(s) to `SENS_I2C_SLAVE_ADDRn` ( $n$ : 0-7). Up to eight slave addresses can be pre-programmed this way. One of these addresses can then be selected for each transaction as part of the RTC I2C instruction.

Once RTC I2C is configured, the main CPU or the ULP coprocessor can communicate with the external I2C devices.

### 28.7.3 Using RTC I2C

#### 28.7.3.1 Instruction Format

The format of RTC I2C instruction is consistent with that of I2C0/I2C1, see Section 17.3.2.2 *CMD\_Controller* in Chapter 17 *I2C Controller*. The only difference is that RTC I2C provides fixed instructions for different operations, as follows:

- Command 0 ~ Command 1: specially for I2C write operation
- Command 2 ~ Command 6: specially for I2C read operation

**Note:** All slave addresses are expressed in 7 bits.

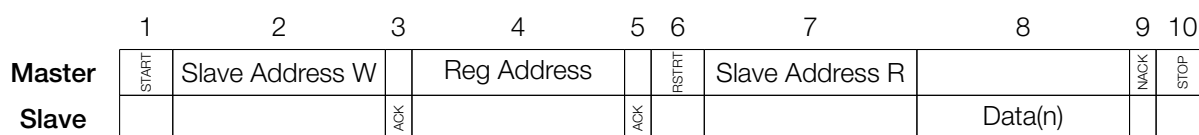
#### 28.7.3.2 I2C\_RD - I2C Read Workflow

Preparation for RTC I2C read:

- Configure the instruction list of RTC I2C (see Section CMD\_Controller in Chapter 17 *I2C Controller*), including instruction order, instruction code, read data number (byte\_num), and other information.
- Configure the slave register address by setting the register `SENS_SAR_I2C_CTRL[18:11]`.
- Start RTC I2C transmission by setting the registers `SENS_SAR_I2C_START_FORCE` and `SENS_SAR_I2C_START`.
- When an `RTC_I2C_RX_DATA_INT` interrupt is received, transfer the read data stored in `RTC_I2C_RDATA` to SRAM RTC slow memory, or use the data directly.

The `I2C_RD` instruction performs the following operations (see Figure 28-25):

1. Master generates a START condition.
2. Master sends slave address, with r/w bit set to 0 ("write"). Slave address is obtained from `SENS_I2C_SLAVE_ADDRn`.
3. Slave generates ACK.
4. Master sends slave register address.
5. Slave generates ACK.
6. Master generates a repeated START (RSTART) condition.
7. Master sends slave address, with r/w bit set to 1 ("read").
8. Slave sends one byte of data.
9. Master checks whether the number of transmitted bytes reaches the number set by the current instruction (byte\_num). If yes, master jumps out of the read instruction and sends an NACK signal. Otherwise master repeats Step 8 and waits for the slave to send the next byte.
10. Master generates a STOP condition and stops reading.



**Figure 28-25. I2C Read Operation**

#### Note:

The RTC I2C peripheral samples the SDA signals on the falling edge of SCL. If the slave changes SDA in less than 0.38 microsecond, the master may receive incorrect data.

### 28.7.3.3 I2C\_WR - I2C Write Workflow

Preparation for RTC I2C write:

- Configure RTC I2C instruction list, including instruction order, instruction code, and the data to be written in byte (byte\_num). See the configuration of I2C0/I2C1 in Section CMD\_Controller in Chapter 17 *I2C Controller*.
- Configure the slave register address by setting the register `SENS_SAR_I2C_CTRL[18:11]`, and the data to be transmitted in `SENS_SAR_I2C_CTRL[26:19]`.

- Set the registers [SENS\\_SAR\\_I2C\\_START\\_FORCE](#) and [SENS\\_SAR\\_I2C\\_START](#) to start the transmission.
- Update the next data to be transmitted in the register [SENS\\_SAR\\_I2C\\_CTRL\[26:19\]](#), each time when an [RTC\\_I2C\\_TX\\_DATA\\_INT](#) interrupt is received.

The I2C\_WR instruction performs the following operations, see Figure 28-26.

1. Master generates a START condition.
2. Master sends slave address, with r/w bit set to 0 (“write”). Slave address is obtained from [SENS\\_I2C\\_SLAVE\\_ADDR<sub>n</sub>](#).
3. Slave generates ACK.
4. Master sends slave register address.
5. Slave generates ACK.
6. Master generates a repeated START condition (RSTART).
7. Master sends slave address, with r/w bit set to 0 (“write”).
8. Master sends one byte of data.
9. Slave generates ACK. Master checks whether the number of transmitted bytes reaches the number set by the current instruction (byte\_num). If yes, master jumps out of the write instruction and starts the next instruction. Otherwise the master repeats step 8 and sends the next byte.
10. Master generates a STOP condition and stops the transmission.

	1	2	3	4	5	6	7	8	9	10
<b>Master</b>	START	Slave Address W		Reg Address		RSTART	Slave Address W	Data(n)		
<b>Slave</b>			ACK		ACK				ACK	STOP

Figure 28-26. I2C Write Operation

### 28.7.3.4 Detecting Error Conditions

Applications can query specific bits in the [RTC\\_I2C\\_INT\\_ST\\_REG](#) register to check if the transaction is successful. To enable checking for specific communication events, their corresponding bits should be set in register [RTC\\_I2C\\_INT\\_ENA\\_REG](#). Note that the bit map is shifted by 1. If a specific communication event is detected and its corresponding bit in register [RTC\\_I2C\\_INT\\_ST\\_REG](#) is set, the event can then be cleared using register [RTC\\_I2C\\_INT\\_CLR\\_REG](#).

### 28.7.4 RTC I2C Interrupts

- [RTC\\_I2C\\_SLAVE\\_TRAN\\_COMP\\_INT](#): Triggered when the slave finishes the transaction.
- [RTC\\_I2C\\_ARBITRATION\\_LOST\\_INT](#): Triggered when the master loses control of the bus.
- [RTC\\_I2C\\_MASTER\\_TRAN\\_COMP\\_INT](#): Triggered when the master completes the transaction.
- [RTC\\_I2C\\_TRANS\\_COMPLETE\\_INT](#): Triggered when a STOP signal is detected.
- [RTC\\_I2C\\_TIME\\_OUT\\_INT](#): Triggered by time out event.
- [RTC\\_I2C\\_ACK\\_ERR\\_INT](#): Triggered by ACK error.

- RTC\_I2C\_RX\_DATA\_INT: Triggered when data is received.
- RTC\_I2C\_TX\_DATA\_INT: Triggered when data is transmitted.
- RTC\_I2C\_DETECT\_START\_INT: Triggered when a START signal is detected.

## 28.8 Base Address

### 28.8.1 ULP Coprocessor Base Address

Users can access ULP coprocessors with two base addresses, which can be seen in Table 158. For more information about accessing peripherals from different buses please see Chapter 1: *System and Memory*.

**Table 158: ULP Coprocessor Base Address**

Module	Bus to Access Peripheral	Base Address
ULP (ALWAYS_ON)	PeriBUS1	0x3F408000
	PeriBUS2	0x60008000
ULP (RTC_PERI)	PeriBUS1	0x3F408800
	PeriBUS2	0x60008800

Wherein:

- ULP (ALWAYS\_ON) represents the registers, which will not be reset due to the power down of RTC\_PERI domain. See Chapter 29 *Low-power Management*.
- ULP (RTC\_PERI) represents the registers in RTC\_PERI domain, which will be reset due to the power down of RTC\_PERI domain. See Chapter 29 *Low-power Management*.

### 28.8.2 RTC I2C Base Address

Users can access the RTC I2C registers in RTC\_PERI domain, including RTC\_PERI registers and I2C registers, with two base addresses, which can be seen in Table 159. For more information about accessing peripherals from different buses please see Chapter 1: *System and Memory*.

**Table 159: RTC I2C Base Address**

Module	Bus to Access Peripheral	Base Address
RTC I2C (RTC_PERI)	PeriBUS1	0x3F408800
	PeriBUS2	0x60008800
RTC I2C (I2C)	PeriBUS1	0x3F408C00
	PeriBUS2	0x60008C00

## 28.9 Register Summary

The address in the following part represents the address offset (relative address) with respect to the base address, not the absolute address. For detailed information about the base address, please refer to Section 28.8.

### 28.9.1 ULP (ALWAYS\_ON) Register Summary

Name	Description	Address	Access
<b>ULP Timer Registers</b>			



Name	Description	Address	Access
<a href="#">RTC_CNTL_ULP_CP_TIMER_REG</a>	Configure the timer	0x00F8	varies
<a href="#">RTC_CNTL_ULP_CP_TIMER_1_REG</a>	Configure sleep cycle of the timer	0x0130	R/W
<b>ULP-FSM Register</b>			
<a href="#">RTC_CNTL_ULP_CP_CTRL_REG</a>	ULP-FSM configuration register	0x00FC	R/W
<b>ULP-RISC-V Register</b>			
<a href="#">RTC_CNTL_COCPU_CTRL_REG</a>	ULP-RISC-V configuration register	0x0100	varies

### 28.9.2 ULP (RTC\_PERI) Register Summary

Name	Description	Address	Access
<b>ULP-RISC-V Registers</b>			
<a href="#">SENS_SAR_COCPU_INT_RAW_REG</a>	Interrupt raw bit of ULP-RISC-V	0x0128	RO
<a href="#">SENS_SAR_COCPU_INT_ENA_REG</a>	Interrupt enable bit of ULP-RISC-V	0x012C	R/W
<a href="#">SENS_SAR_COCPU_INT_ST_REG</a>	Interrupt status bit of ULP-RISC-V	0x0130	RO
<a href="#">SENS_SAR_COCPU_INT_CLR_REG</a>	Interrupt clear bit of ULP-RISC-V	0x0134	WO

### 28.9.3 RTC I2C (RTC\_PERI) Register Summary

Name	Description	Address	Access
<b>RTC I2C Controller Register</b>			
<a href="#">SENS_SAR_I2C_CTRL_REG</a>	Configure RTC I2C transmission	0x0058	R/W
<b>RTC I2C Slave Address Registers</b>			
<a href="#">SENS_SAR_SLAVE_ADDR1_REG</a>	Configure slave addresses 0-1 of RTC I2C	0x0040	R/W
<a href="#">SENS_SAR_SLAVE_ADDR2_REG</a>	Configure slave addresses 2-3 of RTC I2C	0x0044	R/W
<a href="#">SENS_SAR_SLAVE_ADDR3_REG</a>	Configure slave addresses 4-5 of RTC I2C	0x0048	R/W
<a href="#">SENS_SAR_SLAVE_ADDR4_REG</a>	Configure slave addresses 6-7 of RTC I2C	0x004C	R/W

### 28.9.4 RTC I2C (I2C) Register Summary

Name	Description	Address	Access
<b>RTC I2C Signal Setting Registers</b>			
<a href="#">RTC_I2C_SCL_LOW_REG</a>	Configure the low level width of SCL	0x0000	R/W
<a href="#">RTC_I2C_SCL_HIGH_REG</a>	Configure the high level width of SCL	0x0014	R/W
<a href="#">RTC_I2C_SDA_DUTY_REG</a>	Configure the SDA hold time after a negative SCL edge	0x0018	R/W
<a href="#">RTC_I2C_SCL_START_PERIOD_REG</a>	Configure the delay between the SDA and SCL negative edge for a start condition	0x001C	R/W
<a href="#">RTC_I2C_SCL_STOP_PERIOD_REG</a>	Configure the delay between SDA and SCL positive edge for a stop condition	0x0020	R/W
<b>RTC I2C Control Registers</b>			
<a href="#">RTC_I2C_CTRL_REG</a>	Transmission setting	0x0004	R/W
<a href="#">RTC_I2C_STATUS_REG</a>	RTC I2C status	0x0008	RO
<a href="#">RTC_I2C_TO_REG</a>	Configure RTC I2C timeout	0x000C	R/W
<a href="#">RTC_I2C_SLAVE_ADDR_REG</a>	Configure slave address	0x0010	R/W

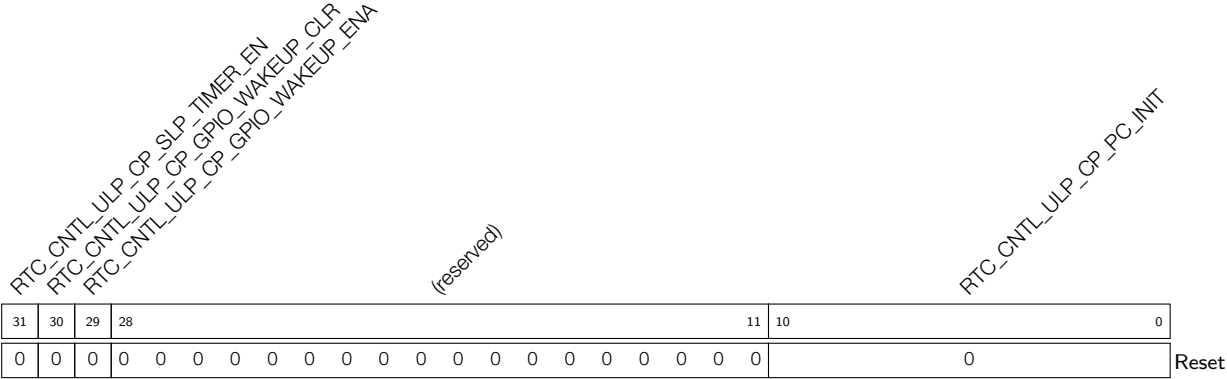
Name	Description	Address	Access
<b>RTC I2C Interrupt Registers</b>			
<a href="#">RTC_I2C_INT_CLR_REG</a>	Clear RTC I2C interrupt	0x0024	WO
<a href="#">RTC_I2C_INT_RAW_REG</a>	RTC I2C raw interrupt	0x0028	RO
<a href="#">RTC_I2C_INT_ST_REG</a>	RTC I2C interrupt status	0x002C	RO
<a href="#">RTC_I2C_INT_ENA_REG</a>	Enable RTC I2C interrupt	0x0030	R/W
<b>RTC I2C Status Register</b>			
<a href="#">RTC_I2C_DATA_REG</a>	RTC I2C read data	0x0034	varies
<b>RTC I2C Command Registers</b>			
<a href="#">RTC_I2C_CMD0_REG</a>	RTC I2C Command 0	0x0038	varies
<a href="#">RTC_I2C_CMD1_REG</a>	RTC I2C Command 1	0x003C	varies
<a href="#">RTC_I2C_CMD2_REG</a>	RTC I2C Command 2	0x0040	varies
<a href="#">RTC_I2C_CMD3_REG</a>	RTC I2C Command 3	0x0044	varies
<a href="#">RTC_I2C_CMD4_REG</a>	RTC I2C Command 4	0x0048	varies
<a href="#">RTC_I2C_CMD5_REG</a>	RTC I2C Command 5	0x004C	varies
<a href="#">RTC_I2C_CMD6_REG</a>	RTC I2C Command 6	0x0050	varies
<a href="#">RTC_I2C_CMD7_REG</a>	RTC I2C Command 7	0x0054	varies
<a href="#">RTC_I2C_CMD8_REG</a>	RTC I2C Command 8	0x0058	varies
<a href="#">RTC_I2C_CMD9_REG</a>	RTC I2C Command 9	0x005C	varies
<a href="#">RTC_I2C_CMD10_REG</a>	RTC I2C Command 10	0x0060	varies
<a href="#">RTC_I2C_CMD11_REG</a>	RTC I2C Command 11	0x0064	varies
<a href="#">RTC_I2C_CMD12_REG</a>	RTC I2C Command 12	0x0068	varies
<a href="#">RTC_I2C_CMD13_REG</a>	RTC I2C Command 13	0x006C	varies
<a href="#">RTC_I2C_CMD14_REG</a>	RTC I2C Command 14	0x0070	varies
<a href="#">RTC_I2C_CMD15_REG</a>	RTC I2C Command 15	0x0074	varies
<b>Version register</b>			
<a href="#">RTC_I2C_DATE_REG</a>	Version control register	0x00FC	R/W

## 28.10 Registers

The address in the following part represents the address offset (relative address) with respect to the base address, not the absolute address. For detailed information about the base address, please refer to [Section 28.8](#).

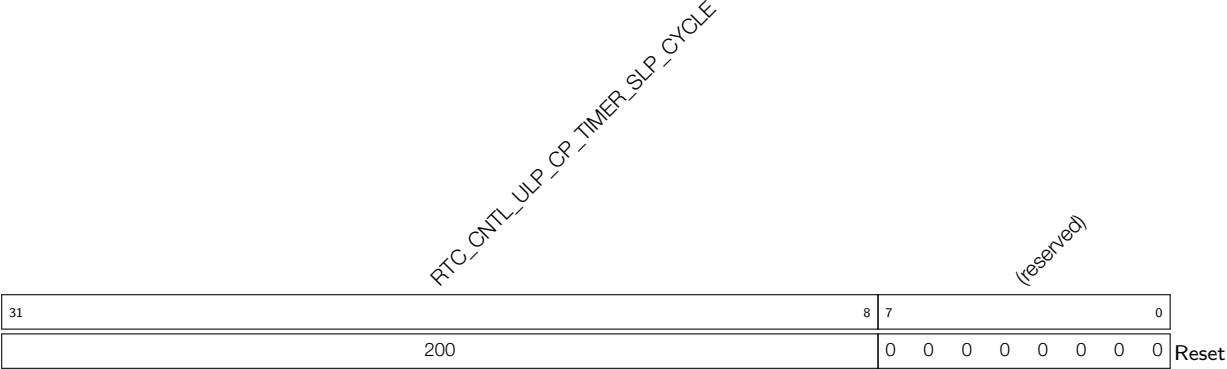
28.10.1 ULP (ALWAYS\_ON) Registers

Register 28.1: RTC\_CNTL\_ULP\_CP\_TIMER\_REG (0x00F8)



- RTC\_CNTL\_ULP\_CP\_PC\_INIT** ULP coprocessor PC initial address. (R/W)
- RTC\_CNTL\_ULP\_CP\_GPIO\_WAKEUP\_ENA** Enable the option of ULP coprocessor woken up by RTC GPIO. (R/W)
- RTC\_CNTL\_ULP\_CP\_GPIO\_WAKEUP\_CLR** Disable the option of ULP coprocessor woken up by RTC GPIO. (WO)
- RTC\_CNTL\_ULP\_CP\_SLP\_TIMER\_EN** ULP coprocessor timer enable bit. 0: Disable hardware timer; 1: Enable hardware timer. (R/W)

Register 28.2: RTC\_CNTL\_ULP\_CP\_TIMER\_1\_REG (0x0130)



- RTC\_CNTL\_ULP\_CP\_TIMER\_SLP\_CYCLE** Set sleep cycles for ULP coprocessor timer. (R/W)

Register 28.3: RTC\_CNTL\_ULP\_CP\_CTRL\_REG (0x00FC)

RTC_CNTL_ULP_CP_START_TOP				(reserved)																												
31	30	29	28	27	0																											0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

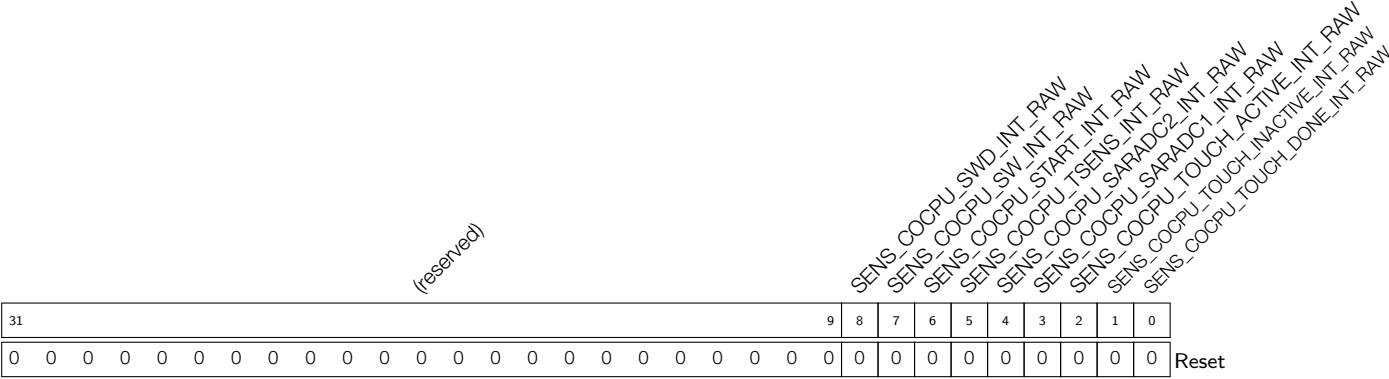
- RTC\_CNTL\_ULP\_CP\_CLK\_FO ULP-FSM clock force on. (R/W)
- RTC\_CNTL\_ULP\_CP\_RESET ULP-FSM clock software reset. (R/W)
- RTC\_CNTL\_ULP\_CP\_FORCE\_START\_TOP Write 1 to start ULP-FSM by software. (R/W)
- RTC\_CNTL\_ULP\_CP\_START\_TOP Write 1 to start ULP-FSM. (R/W)

## 629

ESP32-S2 TRM (Preliminary V0.5)

28.10.2 ULP (RTC\_PERI) Registers

Register 28.5: SENS\_SAR\_COCPU\_INT\_RAW\_REG (0x0128)



- SENS\_COCPU\_TOUCH\_DONE\_INT\_RAW TOUCH\_DONE\_INT interrupt raw bit. (RO)
- SENS\_COCPU\_TOUCH\_INACTIVE\_INT\_RAW TOUCH\_INACTIVE\_INT interrupt raw bit. (RO)
- SENS\_COCPU\_TOUCH\_ACTIVE\_INT\_RAW TOUCH\_ACTIVE\_INT interrupt raw bit. (RO)
- SENS\_COCPU\_SARADC1\_INT\_RAW SARADC1\_DONE\_INT interrupt raw bit. (RO)
- SENS\_COCPU\_SARADC2\_INT\_RAW SARADC2\_DONE\_INT interrupt raw bit. (RO)
- SENS\_COCPU\_TSENS\_INT\_RAW TSENS\_DONE\_INT interrupt raw bit. (RO)
- SENS\_COCPU\_START\_INT\_RAW RISC\_V\_START\_INT interrupt raw bit. (RO)
- SENS\_COCPU\_SW\_INT\_RAW SW\_INT interrupt raw bit. (RO)
- SENS\_COCPU\_SWD\_INT\_RAW SWD\_INT interrupt raw bit. (RO)

Register 28.6: SENS\_SAR\_COCPU\_INT\_ENA\_REG (0x012C)

(reserved)																SENS_COCPU_SWD_INT_ENA			
																SENS_COCPU_SW_INT_ENA			
																SENS_COCPU_START_INT_ENA			
																SENS_COCPU_TSENS_INT_ENA			
																SENS_COCPU_SARADC2_INT_ENA			
																SENS_COCPU_SARADC1_INT_ENA			
																SENS_COCPU_TOUCH_ACTIVE_INT_ENA			
																SENS_COCPU_TOUCH_INACTIVE_INT_ENA			
																SENS_COCPU_TOUCH_DONE_INT_ENA			
31									9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

- SENS\_COCPU\_TOUCH\_DONE\_INT\_ENA TOUCH\_DONE\_INT interrupt enable bit. (R/W)
- SENS\_COCPU\_TOUCH\_INACTIVE\_INT\_ENA TOUCH\_INACTIVE\_INT interrupt enable bit. (R/W)
- SENS\_COCPU\_TOUCH\_ACTIVE\_INT\_ENA TOUCH\_ACTIVE\_INT interrupt enable bit. (R/W)
- SENS\_COCPU\_SARADC1\_INT\_ENA SARADC1\_DONE\_INT interrupt enable bit. (R/W)
- SENS\_COCPU\_SARADC2\_INT\_ENA SARADC2\_DONE\_INT interrupt enable bit. (R/W)
- SENS\_COCPU\_TSENS\_INT\_ENA TSENS\_DONE\_INT interrupt enable bit. (R/W)
- SENS\_COCPU\_START\_INT\_ENA RISC\_V\_START\_INT interrupt enable bit. (R/W)
- SENS\_COCPU\_SW\_INT\_ENA SW\_INT interrupt enable bit. (R/W)
- SENS\_COCPU\_SWD\_INT\_ENA SWD\_INT interrupt enable bit. (R/W)

## 632

ESP32-S2 TRM (Preliminary V0.5)

[Submit Documentation Feedback](#)

**SENS\_COCPU\_SWD\_INT\_ST** **SWD\_INT** interrupt status bit. (RO)



Register 28.8: SENS\_SAR\_COCPU\_INT\_CLR\_REG (0x0134)

(reserved)																								SENS_COCPU_SWD_INT_CLR SENS_COCPU_SW_INT_CLR SENS_COCPU_START_INT_CLR SENS_COCPU_TSNS_INT_CLR SENS_COCPU_SARADC2_INT_CLR SENS_COCPU_SARADC1_INT_CLR SENS_COCPU_TOUCH_INACTIVE_INT_CLR SENS_COCPU_TOUCH_DONE_INT_CLR																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
31																			9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**SENS\_COCPU\_TOUCH\_DONE\_INT\_CLR** [TOUCH\\_DONE\\_INT](#) interrupt clear bit. (WO)

**SENS\_COCPU\_TOUCH\_INACTIVE\_INT\_CLR** [TOUCH\\_INACTIVE\\_INT](#) interrupt clear bit. (WO)

**SENS\_COCPU\_TOUCH\_ACTIVE\_INT\_CLR** [TOUCH\\_ACTIVE\\_INT](#) interrupt clear bit. (WO)

**SENS\_COCPU\_SARADC1\_INT\_CLR** [SARADC1\\_DONE\\_INT](#) interrupt clear bit. (WO)

**SENS\_COCPU\_SARADC2\_INT\_CLR** [SARADC2\\_DONE\\_INT](#) interrupt clear bit. (WO)

**SENS\_COCPU\_TSNS\_INT\_CLR** [TSNS\\_DONE\\_INT](#) interrupt clear bit. (WO)

**SENS\_COCPU\_START\_INT\_CLR** [RISCV\\_START\\_INT](#) interrupt clear bit. (WO)

**SENS\_COCPU\_SW\_INT\_CLR** [SW\\_INT](#) interrupt clear bit. (WO)

**SENS\_COCPU\_SWD\_INT\_CLR** [SWD\\_INT](#) interrupt clear bit. (WO)

### 28.10.3 RTC I2C (RTC\_PERI) Registers

Register 28.9: SENS\_SAR\_I2C\_CTRL\_REG (0x0058)

(reserved)					SENS_SAR_I2C_CTRL																
31	30	29	28	27																	
0	0	0	0		0																

**SENS\_SAR\_I2C\_CTRL** RTC I2C control data; active only when [SENS\\_SAR\\_I2C\\_START\\_FORCE](#) = 1. (R/W)

**SENS\_SAR\_I2C\_START** Start RTC I2C; active only when [SENS\\_SAR\\_I2C\\_START\\_FORCE](#) = 1. (R/W)

**SENS\_SAR\_I2C\_START\_FORCE** 0: RTC I2C started by FSM; 1: RTC I2C started by software. (R/W)

Register 28.10: SENS\_SAR\_SLAVE\_ADDR1\_REG (0x0040)

(reserved)										SENS_I2C_SLAVE_ADDR0												SENS_I2C_SLAVE_ADDR1													
31											22	21											11	10											0
0	0	0	0	0	0	0	0	0	0	0	0x0										0x0										Reset				

**SENS\_I2C\_SLAVE\_ADDR1** RTC I2C slave address 1. (R/W)

**SENS\_I2C\_SLAVE\_ADDR0** RTC I2C slave address 0. (R/W)

Register 28.11: SENS\_SAR\_SLAVE\_ADDR2\_REG (0x0044)

(reserved)										SENS_I2C_SLAVE_ADDR2												SENS_I2C_SLAVE_ADDR3													
31											22	21											11	10											0
0	0	0	0	0	0	0	0	0	0	0	0x0										0x0										Reset				

**SENS\_I2C\_SLAVE\_ADDR3** RTC I2C slave address 3. (R/W)

**SENS\_I2C\_SLAVE\_ADDR2** RTC I2C slave address 2. (R/W)

Register 28.12: SENS\_SAR\_SLAVE\_ADDR3\_REG (0x0048)

(reserved)										SENS_I2C_SLAVE_ADDR4											SENS_I2C_SLAVE_ADDR5														
31											22	21											11	10											0
0	0	0	0	0	0	0	0	0	0	0	0x0										0x0										Reset				

**SENS\_I2C\_SLAVE\_ADDR5** RTC I2C slave address 5. (R/W)

**SENS\_I2C\_SLAVE\_ADDR4** RTC I2C slave address 4. (R/W)

**Register 28.13: SENS\_SAR\_SLAVE\_ADDR4\_REG (0x004C)**

(reserved)										SENS_I2C_SLAVE_ADDR6										SENS_I2C_SLAVE_ADDR7															
31											22	21											11	10											0
0	0	0	0	0	0	0	0	0	0	0	0x0										0x0										Reset				

**SENS\_I2C\_SLAVE\_ADDR7** RTC I2C slave address 7. (R/W)

**SENS\_I2C\_SLAVE\_ADDR6** RTC I2C slave address 6. (R/W)

### 28.10.4 RTC I2C (I2C) Registers

**Register 28.14: RTC\_I2C\_SCL\_LOW\_REG (0x0000)**

(reserved)												RTC_I2C_SCL_LOW_PERIOD_REG																																																																																																											
31												20												19																			0																																																																												
0												0												0												0												0												0												0												0												0x100												Reset											

**RTC\_I2C\_SCL\_LOW\_PERIOD\_REG** This register is used to configure how many clock cycles SCL remains low. (R/W)

**Register 28.15: RTC\_I2C\_SCL\_HIGH\_REG (0x0014)**

(reserved)													RTC_I2C_SCL_HIGH_PERIOD_REG																	
3120													190																	
00000000000000													0x100Reset																	

**RTC\_I2C\_SCL\_HIGH\_PERIOD\_REG** This register is used to configure how many cycles SCL remains high. (R/W)

### Register 28.16: RTC\_I2C\_SDA\_DUTY\_REG (0x0018)

(reserved)												RTC_I2C_SDA_DUTY_NUM																															
31												20												19																			0
0 0 0 0 0 0 0 0 0 0 0 0												0x010																			Reset												

<b>RTC_I2C_SDA_DUTY_NUM</b>	The number of clock cycles between the SDA switch and the falling edge of SCL. (R/W)
-----------------------------	--

### Register 28.17: RTC\_I2C\_SCL\_START\_PERIOD\_REG (0x001C)

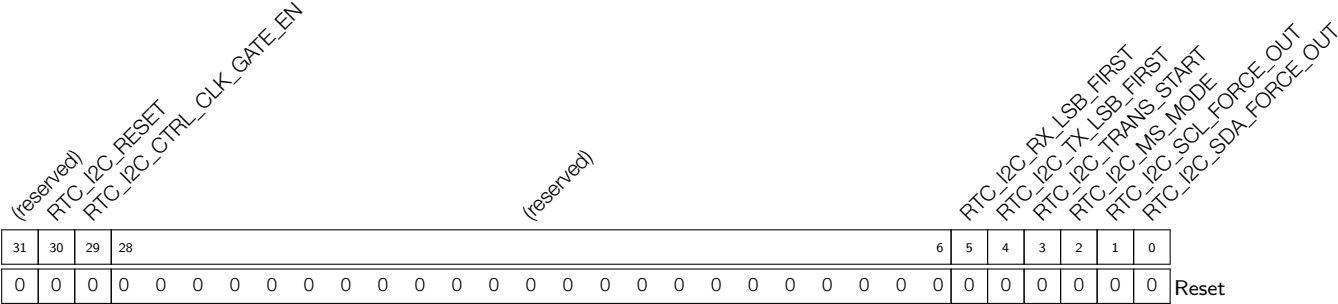
**RTC\_I2C\_SCL\_START\_PERIOD** Number of clock cycles to wait after generating a start condition.  
(R/W)

### Register 28.18: RTC\_I2C\_SCL\_STOP\_PERIOD\_REG (0x0020)

(reserved)												RTC_I2C_SCL_STOP_PERIOD																																																																																																																							
31												20												19												0																																																																																															
0												0												0												0												0												0												0												0												0												8												Reset											

**RTC\_I2C\_SCL\_STOP\_PERIOD** Number of clock cycles to wait before generating a stop condition.  
(R/W)

Register 28.19: RTC\_I2C\_CTRL\_REG (0x0004)



- RTC\_I2C\_SDA\_FORCE\_OUT** SDA output mode. 0: open drain; 1: push pull. (R/W)
- RTC\_I2C\_SCL\_FORCE\_OUT** SCL output mode. 0: open drain; 1: push pull. (R/W)
- RTC\_I2C\_MS\_MODE** Set this bit to configure RTC I2C as a master. (R/W)
- RTC\_I2C\_TRANS\_START** Set this bit to 1, RTC I2C starts sending data. (R/W)
- RTC\_I2C\_TX\_LSB\_FIRST** This bit is used to control the sending mode. 0: send data from the most significant bit; 1: send data from the least significant bit. (R/W)
- RTC\_I2C\_RX\_LSB\_FIRST** This bit is used to control the storage mode for received data. 0: receive data from the most significant bit; 1: receive data from the least significant bit. (R/W)
- RTC\_I2C\_CTRL\_CLK\_GATE\_EN** RTC I2C controller clock gate. (R/W)
- RTC\_I2C\_RESET** RTC I2C software reset. (R/W)

**Register 28.20: RTC\_I2C\_STATUS\_REG (0x0008)**

(reserved)																								RTC_I2C_OP_CNT								RTC_I2C_BYTE_TRANS								RTC_I2C_SLAVE_ADDRESSED								RTC_I2C_BUS_BUSY								RTC_I2C_ARB_LOST								RTC_I2C_SLAVE_RW								RTC_I2C_ACK_REC																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
31																								8								7	6	5	4	3	2	1	0	Reset																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
0																								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**RTC\_I2C\_ACK\_REC** The received ACK value. 0: ACK; 1: NACK. (RO)

**RTC\_I2C\_SLAVE\_RW** 0: master writes to slave; 1: master reads from slave. (RO)

**RTC\_I2C\_ARB\_LOST** When the RTC I2C loses control of SCL line, the register changes to 1. (RO)

**RTC\_I2C\_BUS\_BUSY** 0: RTC I2C bus is in idle state; 1: RTC I2C bus is busy transferring data. (RO)

**RTC\_I2C\_SLAVE\_ADDRESSED** When the address sent by the master matches the address of the slave, then this bit will be set. (RO)

**RTC\_I2C\_BYTE\_TRANS** This field changes to 1 when one byte is transferred. (RO)

**RTC\_I2C\_OP\_CNT** Indicate which operation is working. (RO)

**Register 28.21: RTC\_I2C\_TO\_REG (0x000C)**

(reserved)												RTC_I2C_TIME_OUT_REG																															
31												20												19																			0
0 0 0 0 0 0 0 0 0 0 0 0												0x10000																			Reset												

**RTC\_I2C\_TIME\_OUT\_REG** Timeout threshold. (R/W)

Register 28.22: RTC\_I2C\_SLAVE\_ADDR\_REG (0x0010)

RTC_I2C_ADDR_10BIT_EN																															(reserved)																			
31	30																																																	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset																

**RTC\_I2C\_ADDR\_10BIT\_EN** This field is used to enable the slave 10-bit addressing mode. (R/W)

Register 28.23: RTC\_I2C\_INT\_CLR\_REG (0x0024)

(reserved)																				RTC_I2C_DETECT_START_INT_CLR RTC_I2C_TX_DATA_INT_CLR RTC_I2C_RX_DATA_INT_CLR RTC_I2C_ACK_ERR_INT_CLR RTC_I2C_TIME_OUT_INT_CLR RTC_I2C_TRANS_COMPLETE_INT_CLR RTC_I2C_MASTER_TRAN_COMP_INT_CLR RTC_I2C_ARBITRATION_LOST_INT_CLR RTC_I2C_SLAVE_TRAN_COMP_INT_CLR																			
31																				9	8	7	6	5	4	3	2	1	0										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset						

**RTC\_I2C\_SLAVE\_TRAN\_COMP\_INT\_CLR** [RTC\\_I2C\\_SLAVE\\_TRAN\\_COMP\\_INT](#) interrupt clear bit. (WO)

**RTC\_I2C\_ARBITRATION\_LOST\_INT\_CLR** [RTC\\_I2C\\_ARBITRATION\\_LOST\\_INT](#) interrupt clear bit. (WO)

**RTC\_I2C\_MASTER\_TRAN\_COMP\_INT\_CLR** [RTC\\_I2C\\_MASTER\\_TRAN\\_COMP\\_INT](#) interrupt clear bit. (WO)

**RTC\_I2C\_TRANS\_COMPLETE\_INT\_CLR** [RTC\\_I2C\\_TRANS\\_COMPLETE\\_INT](#) interrupt clear bit. (WO)

**RTC\_I2C\_TIME\_OUT\_INT\_CLR** [RTC\\_I2C\\_TIME\\_OUT\\_INT](#) interrupt clear bit. (WO)

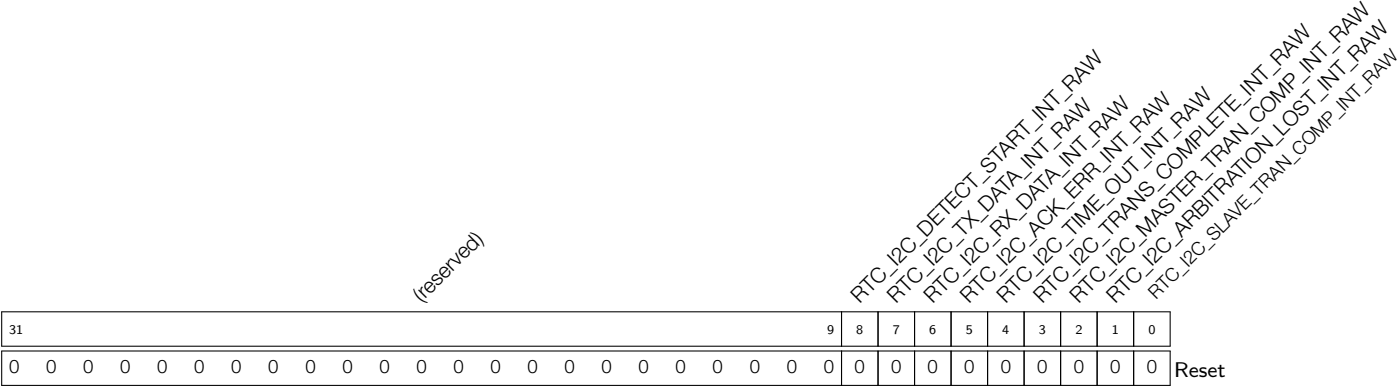
**RTC\_I2C\_ACK\_ERR\_INT\_CLR** [RTC\\_I2C\\_ACK\\_ERR\\_INT](#) interrupt clear bit. (WO)

**RTC\_I2C\_RX\_DATA\_INT\_CLR** [RTC\\_I2C\\_RX\\_DATA\\_INT](#) interrupt clear bit. (WO)

**RTC\_I2C\_TX\_DATA\_INT\_CLR** [RTC\\_I2C\\_TX\\_DATA\\_INT](#) interrupt clear bit. (WO)

**RTC\_I2C\_DETECT\_START\_INT\_CLR** [RTC\\_I2C\\_DETECT\\_START\\_INT](#) interrupt clear bit. (WO)

Register 28.24: RTC\_I2C\_INT\_RAW\_REG (0x0028)



**RTC\_I2C\_SLAVE\_TRAN\_COMP\_INT\_RAW** [RTC\\_I2C\\_SLAVE\\_TRAN\\_COMP\\_INT](#) interrupt raw bit. (RO)

**RTC\_I2C\_ARBITRATION\_LOST\_INT\_RAW** [RTC\\_I2C\\_ARBITRATION\\_LOST\\_INT](#) interrupt raw bit. (RO)

**RTC\_I2C\_MASTER\_TRAN\_COMP\_INT\_RAW** [RTC\\_I2C\\_MASTER\\_TRAN\\_COMP\\_INT](#) interrupt raw bit. (RO)

**RTC\_I2C\_TRANS\_COMPLETE\_INT\_RAW** [RTC\\_I2C\\_TRANS\\_COMPLETE\\_INT](#) interrupt raw bit. (RO)

**RTC\_I2C\_TIME\_OUT\_INT\_RAW** [RTC\\_I2C\\_TIME\\_OUT\\_INT](#) interrupt raw bit. (RO)

**RTC\_I2C\_ACK\_ERR\_INT\_RAW** [RTC\\_I2C\\_ACK\\_ERR\\_INT](#) interrupt raw bit. (RO)

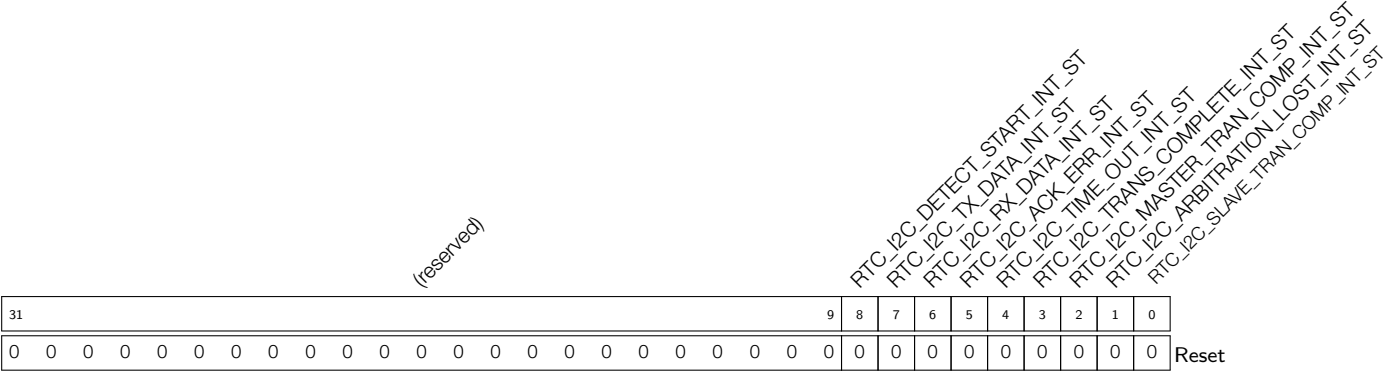
**RTC\_I2C\_RX\_DATA\_INT\_RAW** [RTC\\_I2C\\_RX\\_DATA\\_INT](#) interrupt raw bit. (RO)

**RTC\_I2C\_TX\_DATA\_INT\_RAW** [RTC\\_I2C\\_TX\\_DATA\\_INT](#) interrupt raw bit. (RO)

**RTC\_I2C\_DETECT\_START\_INT\_RAW** [RTC\\_I2C\\_DETECT\\_START\\_INT](#) interrupt raw bit. (RO)



Register 28.25: RTC\_I2C\_INT\_ST\_REG (0x002C)



**RTC\_I2C\_SLAVE\_TRAN\_COMP\_INT\_ST** [RTC\\_I2C\\_SLAVE\\_TRAN\\_COMP\\_INT](#) interrupt status bit. (RO)

**RTC\_I2C\_ARBITRATION\_LOST\_INT\_ST** [RTC\\_I2C\\_ARBITRATION\\_LOST\\_INT](#) interrupt status bit. (RO)

**RTC\_I2C\_MASTER\_TRAN\_COMP\_INT\_ST** [RTC\\_I2C\\_MASTER\\_TRAN\\_COMP\\_INT](#) interrupt status bit. (RO)

**RTC\_I2C\_TRANS\_COMPLETE\_INT\_ST** [RTC\\_I2C\\_TRANS\\_COMPLETE\\_INT](#) interrupt status bit. (RO)

**RTC\_I2C\_TIME\_OUT\_INT\_ST** [RTC\\_I2C\\_TIME\\_OUT\\_INT](#) interrupt status bit. (RO)

**RTC\_I2C\_ACK\_ERR\_INT\_ST** [RTC\\_I2C\\_ACK\\_ERR\\_INT](#) interrupt status bit. (RO)

**RTC\_I2C\_RX\_DATA\_INT\_ST** [RTC\\_I2C\\_RX\\_DATA\\_INT](#) interrupt status bit. (RO)

**RTC\_I2C\_TX\_DATA\_INT\_ST** [RTC\\_I2C\\_TX\\_DATA\\_INT](#) interrupt status bit. (RO)

**RTC\_I2C\_DETECT\_START\_INT\_ST** [RTC\\_I2C\\_DETECT\\_START\\_INT](#) interrupt status bit. (RO)

Register 28.26: RTC\_I2C\_INT\_ENA\_REG (0x0030)

(reserved)																								RTC_I2C_DETECT_START_INT_ENA RTC_I2C_TX_DATA_INT_ENA RTC_I2C_RX_DATA_INT_ENA RTC_I2C_ACK_ERR_INT_ENA RTC_I2C_TIME_OUT_INT_ENA RTC_I2C_TRANS_COMPLETE_INT_ENA RTC_I2C_MASTER_TRAN_COMP_INT_ENA RTC_I2C_ARBITRATION_LOST_INT_ENA RTC_I2C_SLAVE_TRAN_COMP_INT_ENA																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
31																								9	8	7	6	5	4	3	2	1	0	Reset																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**RTC\_I2C\_SLAVE\_TRAN\_COMP\_INT\_ENA** [RTC\\_I2C\\_SLAVE\\_TRAN\\_COMP\\_INT](#) interrupt enable bit. (R/W)

**RTC\_I2C\_ARBITRATION\_LOST\_INT\_ENA** [RTC\\_I2C\\_ARBITRATION\\_LOST\\_INT](#) interrupt enable bit. (R/W)

**RTC\_I2C\_MASTER\_TRAN\_COMP\_INT\_ENA** [RTC\\_I2C\\_MASTER\\_TRAN\\_COMP\\_INT](#) interrupt enable bit. (R/W)

**RTC\_I2C\_TRANS\_COMPLETE\_INT\_ENA** [RTC\\_I2C\\_TRANS\\_COMPLETE\\_INT](#) interrupt enable bit. (R/W)

**RTC\_I2C\_TIME\_OUT\_INT\_ENA** [RTC\\_I2C\\_TIME\\_OUT\\_INT](#) interrupt enable bit. (R/W)

**RTC\_I2C\_ACK\_ERR\_INT\_ENA** [RTC\\_I2C\\_ACK\\_ERR\\_INT](#) interrupt enable bit. (R/W)

**RTC\_I2C\_RX\_DATA\_INT\_ENA** [RTC\\_I2C\\_RX\\_DATA\\_INT](#) interrupt enable bit. (R/W)

**RTC\_I2C\_TX\_DATA\_INT\_ENA** [RTC\\_I2C\\_TX\\_DATA\\_INT](#) interrupt enable bit. (R/W)

**RTC\_I2C\_DETECT\_START\_INT\_ENA** [RTC\\_I2C\\_DETECT\\_START\\_INT](#) interrupt enable bit. (R/W)

Register 28.27: RTC\_I2C\_DATA\_REG (0x0034)

RTC_I2C_DONE																(reserved)																RTC_I2C_SLAVE_TX_DATA																RTC_I2C_RDATA															
31	30															16	15	8															7	0																													
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0																0x0																Reset															

**RTC\_I2C\_RDATA** Data received. (RO)

**RTC\_I2C\_SLAVE\_TX\_DATA** The data sent by slave. (R/W)

**RTC\_I2C\_DONE** RTC I2C transmission is done. (RO)



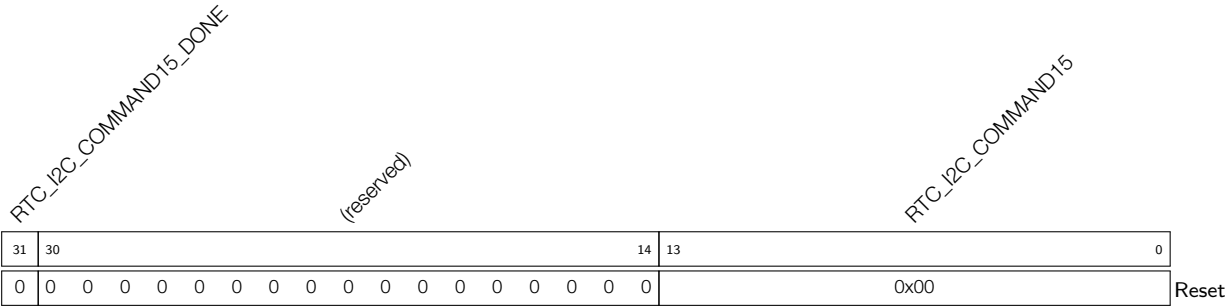








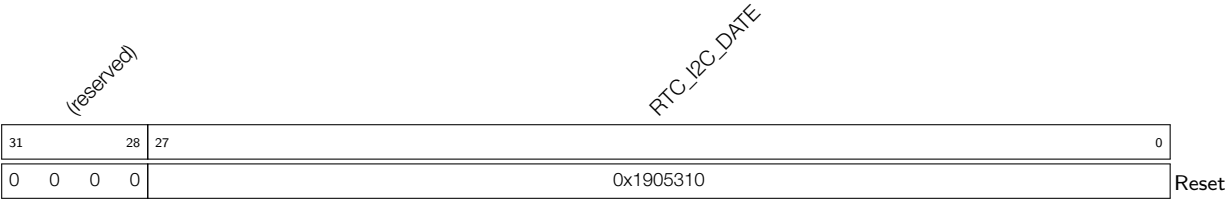
Register 28.43: RTC\_I2C\_CMD15\_REG (0x0074)



**RTC\_I2C\_COMMAND15** Content of command 15. For more information, please refer to the register [I2C\\_COMD15\\_REG](#) in Chapter I2C Controller. (R/W)

**RTC\_I2C\_COMMAND15\_DONE** When command 15 is done, this bit changes to 1. (RO)

Register 28.44: RTC\_I2C\_DATE\_REG (0x00FC)



**RTC\_I2C\_DATE** Version control register (R/W)



## 29. Low-power Management

### 29.1 Introduction

ESP32-S2 has an advanced Power Management Unit (PMU), which can flexibly power up different power domains of the chip, to achieve the best balance among chip performance, power consumption, and wakeup latency. To simplify power management for typical scenarios, ESP32-S2 has predefined five power modes, which are preset configurations that power up different combinations of power domains. On top of that, the chip also allows the users to independently power up any particular power domain to meet more complex requirements. ESP32-S2 has integrated two Ultra-Low-Power co-processors (ULP co-processors), which allow the chip to work when most of the power domains are powered down, thus achieving extremely low-power consumption.

### 29.2 Features

ESP32-S2's low-power management supports the following features:

- ULP co-processors supported in all power modes
- Five predefined power modes to simplify power management for typical scenarios
- Up to 16 KB of retention memory
- 8 x 32-bit retention registers
- RTC Boot supported for reduced wakeup latency

In this chapter, we first introduce the working process of ESP32-S2's low-power management, then introduce the predefined power modes of the chip, and at last, introduce the RTC boot of the chip.

### 29.3 Functional Description

ESP32-S2's low-power management involves the following components:

- Power management unit: controls the power supply to Analog, RTC and Digital power domains.
- Power isolation unit: isolates different power domains, so any powered down power domain does not affect the powered up ones.
- Low-power clocks: provide clocks to power domains working in low-power modes.
- Timers:
  - RTC timer: logs the status of the RTC main state machine in dedicated registers.
  - ULP timer: wakes up the ULP co-processors at a predefined time. For details, please refer to [Chapter 28 ULP Coprocessor](#).
  - Touch sensor timer: wakes up the touch sensor at a predefined time.
- 8 x 32-bit “always-on” retention registers: These registers are always powered up and can be used for storing data that cannot be lost.
- 22 x “always-on” pins: These pins are always powered up and can be used as wakeup sources when the chip is working in the low-power modes (for details, please refer to [Section 29.4.4](#)), or can be used as regular GPIOs (for details, please refer to [Chapter 5 IO MUX and GPIO Matrix](#)).

- RTC slow memory: supports 8 KB SRAM, which can be used as retention memory or to store ULP directives and data.
- RTC fast memory: supports 8 KB SRAM, which can be used as retention memory.
- Regulators: regulate the power supply to different power domains.

The schematic diagram of ESP32-S2's low-power management is shown in Figure 29-1.

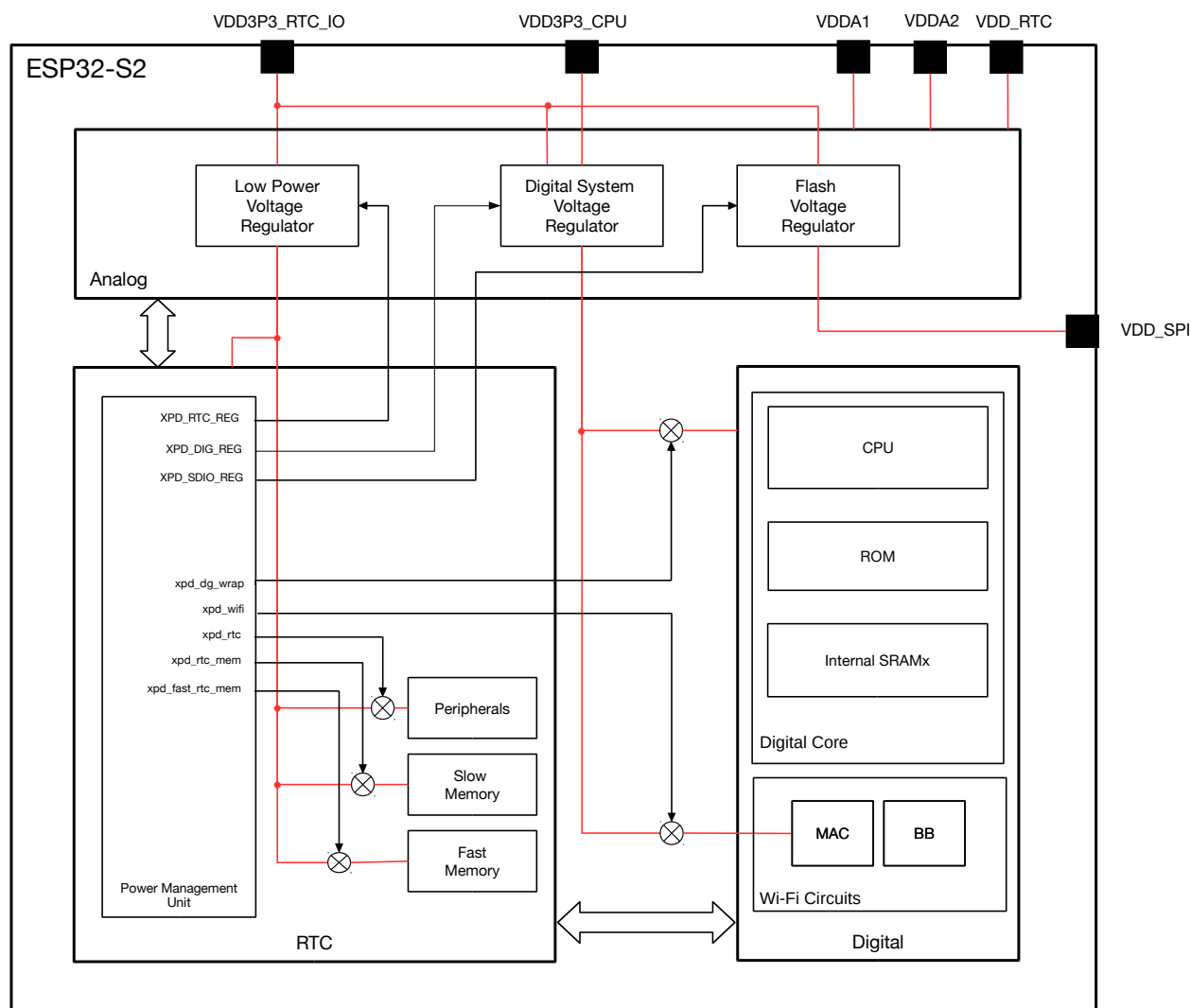


Figure 29-1. Low-power Management Schematics

### 29.3.1 Power Management Unit

ESP32-S2's power management unit controls the power supply to different power domains. The main components of the power management unit include:

- RTC main state machine: generates power gating, clock gating, and reset signals.
- Power controllers: power up and power down different power domains, according to the power gating signals from the main state machine.
- Sleep / wakeup controllers: send sleep or wakeup requests to the RTC main state machine.
- Clock controller: selects and powers up / down clock sources.

In ESP32-S2's power management unit, the sleep / wakeup controllers send sleep or wakeup requests to the RTC main state machine, which then generates power gating, clock gating, and reset signals. Then, the power controller and clock controller power up and power down different power domains and clock sources, according to the signals generated by the RTC main state machine, so that the chip enters or exits the low-power modes. The main workflow is shown in Figure 29-2.

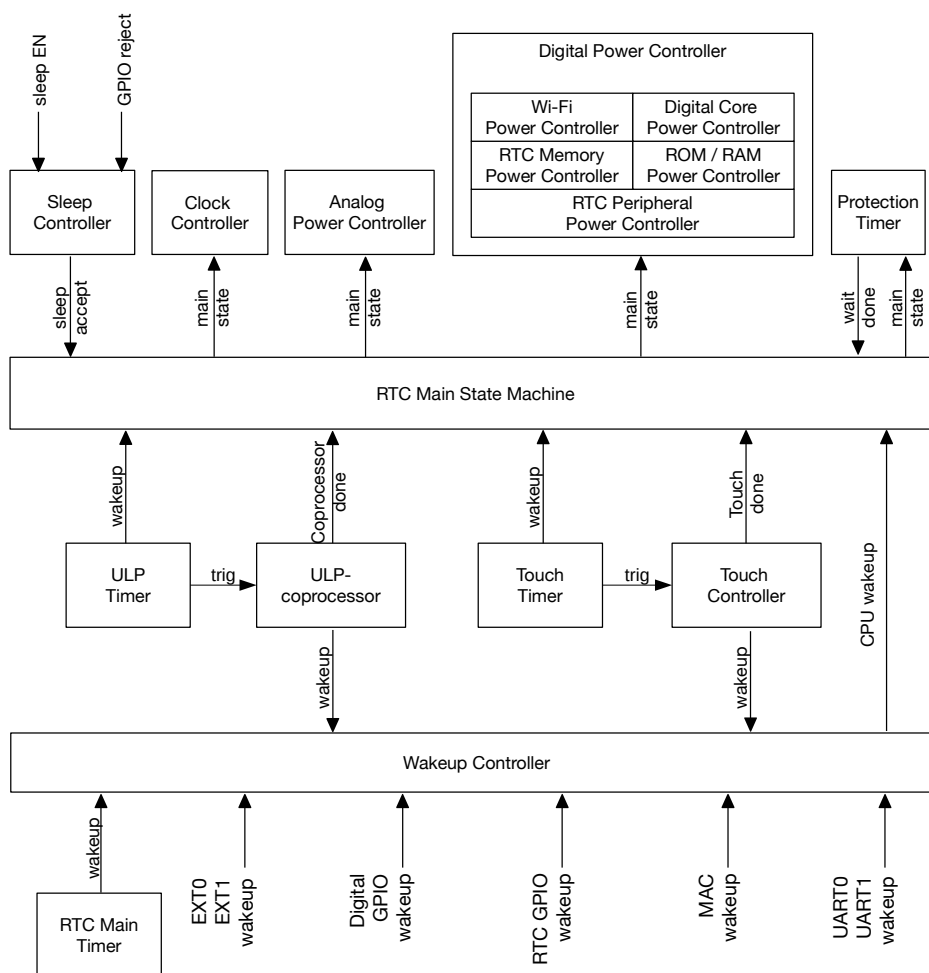


Figure 29-2. Power Management Unit Workflow

**Note:**

For more detailed description about power domains, please refer to 29.4.2.

### 29.3.2 Low-Power Clocks

In general, ESP32-S2 powers down its 40 MHz crystal oscillator and PLL to reduce power consumption when working in low-power modes. During this time, the chip's low-power clocks remain on to provide clocks to different power domains, such as the power management unit, RTC peripherals, RTC fast memory, RTC slow memory, and Wi-Fi digital circuits in the digital domain.

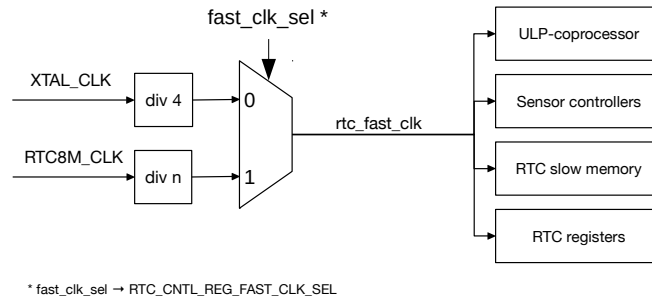


Figure 29-3. Low-Power Clocks for RTC Power Domains

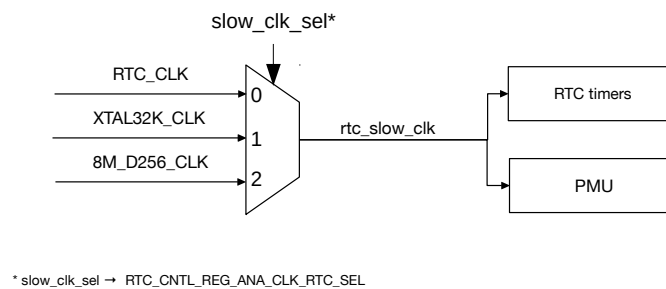


Figure 29-4. Low-Power Clocks for RTC Power Domains

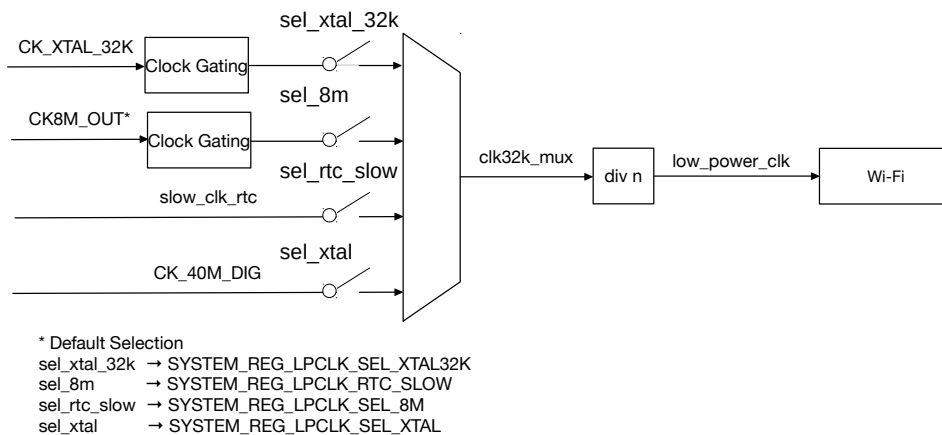


Figure 29-5. Low-Power Clocks for RTC Fast Memory and Slow Memory

Table 164: Low-Power Clocks

Clock Type	Clock Source	Selection Option	Power Domain
RTC fast clock	RTC8M_CLK divided by n <sup>1</sup>	<a href="#">RTC_CNTL_FAST_CLK_RTC_SEL</a>	RTC peripherals
	XTAL_CLK divided by 4		RTC fast memory
			RTC slow memory
RTC slow clock	XTAL32K_CLK	<a href="#">RTC_CNTL_ANA_CLK_RTC_SEL</a>	Power management unit
	RTC8M_D256_CLK		
	RTC_CLK <sup>2</sup>		
Low-Power clocks	XTAL32K_CLK	<a href="#">SYSTEM_LPCLK_SEL_XTAL32K</a>	Digital system (Wi-Fi) in low-power modes
	RTC8M_CLK	<a href="#">SYSTEM_LPCLK_SEL_8M</a>	
	RTC_CLK	<a href="#">SYSTEM_LPCLK_RTC_SLOW</a>	
	XTAL_CLK	<a href="#">SYSTEM_LPCLK_SEL_XTAL</a>	

**Note:**

1. The default RTC fast clock source.
2. The default RTC slow clock source.

For more detailed description about clocks, please refer to [2 Reset and Clock](#).

### 29.3.3 Timers

ESP32-S2's low-power management uses 3 timers:

- RTC timer
- ULP timer
- Touch sensor timer

This section only introduces the RTC timer. For detailed description of ULP timer and touch sensor timer, please refer to Chapters [28 ULP Coprocessor](#).

The readable 48-bit RTC timer is a real-time counter (using RTC slow clock) that can be configured to log the time when one of the following events happens. For details, see [Table 165](#).

Table 165: The Triggering Conditions for the RTC Timer

Enabling Options	Descriptions
<a href="#">RTC_CNTL_TIMER_XTL_OFF</a>	1. RTC main state machine powers down; 2. 40 MHz crystal powers up.
<a href="#">RTC_CNTL_TIMER_SYS_STALL</a>	CPU enters or exits the stall state. This is to ensure the SYS_TIMER is continuous in time.
<a href="#">RTC_CNTL_TIMER_SYS_RST</a>	Resetting digital system completes.
<a href="#">RTC_CNTL_TIME_UPDATE</a>	Register <a href="#">RTC_CNTL_RTC_TIME_UPDATE</a> is configured by CPU (i.e. users).

The RTC timer updates two groups of registers upon any new trigger. The first group logs the information of the current trigger, and the other logs the previous trigger. Detailed information about these two register groups is

shown below:

- Register group 0: logs the status of RTC timer at the current trigger.
  - [RTC\\_CNTL\\_TIME\\_HIGH0\\_REG](#)
  - [RTC\\_CNTL\\_TIME\\_LOW0\\_REG](#)
- Register group 1: logs the status of RTC timer at the previous trigger.
  - [RTC\\_CNTL\\_TIME\\_HIGH1\\_REG](#)
  - [RTC\\_CNTL\\_TIME\\_LOW1\\_REG](#)

On a new trigger, information on previous trigger is moved from register group 0 to register group 1 (and the original trigger logged in register group 1 is overrode), and this new trigger is logged in register group 0. Therefore, only the last two triggers can be logged at any time.

It should be noted that any reset / sleep other than power-up reset will not stop or reset the RTC timer.

Also, the RTC timer can be used as a wakeup source. For details, see Section [29.4.4](#).

### 29.3.4 Regulators

ESP32-S2 has three regulators to regulate the power supply to different power domains:

- Digital system voltage regulator for digital power domains;
- Low-power voltage regulator for RTC power domains;
- Flash voltage regulator for the rest of power domains.

**Note:**

For more detailed description about power domains, please refer to Section [29.4.2](#).

#### 29.3.4.1 Digital System Voltage Regulator

ESP32-S2's built-in digital system voltage regulator converts the external power supply (typically 3.3 V) to 1.1 V for digital power domains. This regulator has an adjustable output. For the architecture of the ESP32-S2 digital system voltage regulator, see Figure [29-6](#).

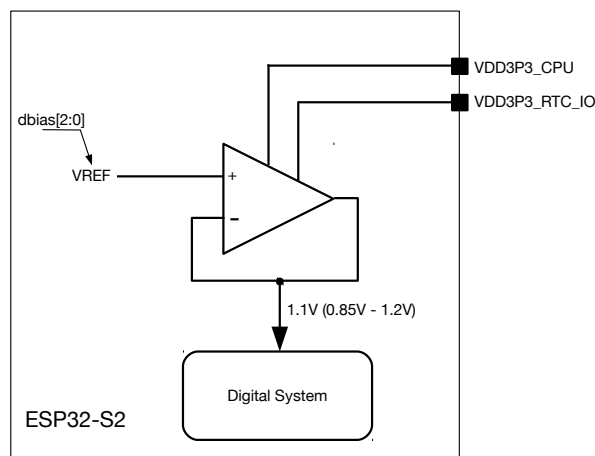


Figure 29-6. Digital System Regulator

1. When `XPD_DIG_REG == 1`, the regulator outputs a 1.1 V voltage and the power domains in digital system are able to run; when `XPD_DIG_REG == 0`, both the regulator and the power domains in digital system stop running.
2. `DIG_DBIAS[2:0]` tunes the supply voltage of the digital system:

$$VDD\_DIG = 0.90 + DBIAS \times 0.05V$$

3. The current to power domains in digital system comes from pin `VDD3P3_CPU` and pin `VDD3P3_RTC_IO`.

#### 29.3.4.2 Low-power Voltage Regulator

ESP32-S2's built-in low-power voltage regulator converts the external power supply (typically 3.3 V) to 1.1 V for RTC power domains. This regulator has an adjustable output to achieve lower power consumption and can output even lower voltage in Deep-sleep mode and Hibernation mode. For the architecture of the ESP32-S2 low-power voltage regulator, see Figure 29-1.

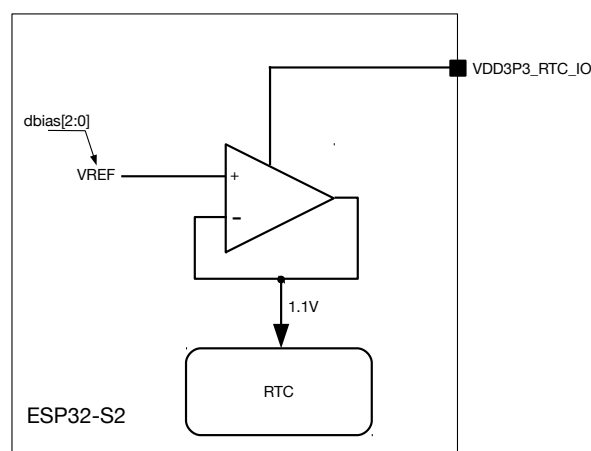


Figure 29-7. Low-power voltage regulator

1. When the pin CHIP\_PU is at a high level, the low-power voltage regulator cannot be turned off, but only switching between normal-work mode and Deep-sleep mode.
2. RTC\_DBIAS[2:0] can be used to tune the output voltage:

$$VDD\_RTC = 0.90 + DBIAS \times 0.05V$$

3. The current to the RTC power domains comes from pin VDD3P3\_RTC\_IO.

### 29.3.4.3 Flash Voltage Regulator

ESP32-S2's built-in flash voltage regulator can supply a voltage of 3.3 V or 1.8 V to other components outside of digital system and RTC, such as flash. For the architecture of the ESP32-S2 flash voltage regulator, see Figure 29-8.

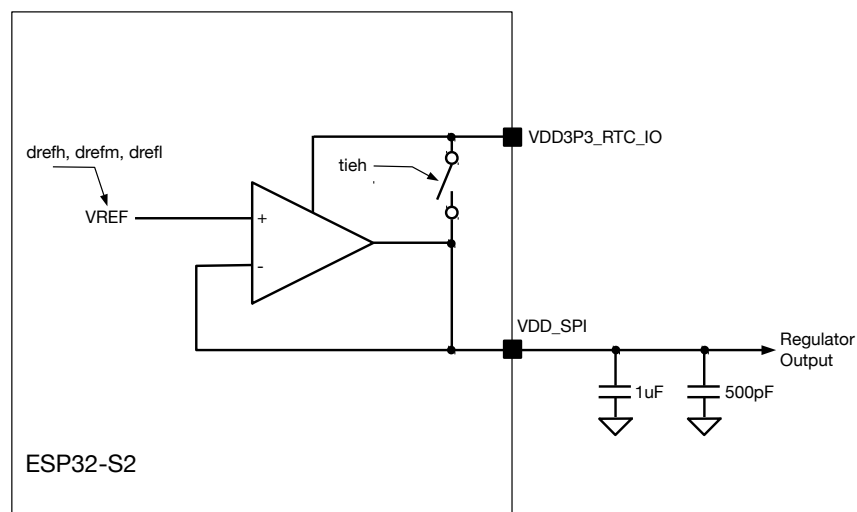


Figure 29-8. Flash voltage regulator

1. When XPD\_SDIO\_VREG == 1, the regulator outputs a voltage of 3.3 V or 1.8 V; when XPD\_SDIO\_VREG == 0, the output is high-impedance. In this case, the voltage is provided by the external power supply.
2. When SDIO\_TIEH == 1, the regulator shorts pin VDD\_SDIO to pin VDD3P3\_RTC, and outputs a voltage of 3.3 V, which is the voltage of pin VDD3P3\_RTC. When SDIO\_TIEH == 0, the regulator outputs the reference voltage VREF, which is typically 1.8 V.
3. DREFH\_SDIO, DREFM\_SDIO and DREFL\_SDIO could be used to fine tune the reference voltage VREF, but are not recommended because this fine tuning may jeopardize the stability of the inner loop.
4. When the regulator output is 3.3 V or 1.8 V, the output current comes from the pin VDD3P3\_RTC\_IO.

The flash voltage regulator can be configured via RTC registers or eFuse controller registers.

- The configuration of XPD\_SDIO\_VREG:
  - When the chip is in active mode, RTC\_CNTL\_SDIO\_FORCE == 0 and VDD\_SPI\_FORCE(EFUSE) == 1, the XPD\_SDIO\_VREG voltage is defined by XPD\_VDD\_SPI\_REG(EFUSE);
  - When the chip is in sleep modes and RTC\_CNTL\_SDIO\_PD\_EN == 1, XPD\_SDIO\_VREG is 0;



- When `RTC_CNTL_SDIO_FORCE == 1`, `XPD_SDIO_VREG` is defined by `RTC_CNTL_XPD_SDIO_REG`.
- The configuration of `SDIO_TIEH`:
  - When `RTC_CNTL_SDIO_FORCE == 0` and `VDD_SPI_FORCE(EFUSE) == 1`, `SDIO_TIEH = VDD_SDIO_TIEH(EFUSE)`
  - Otherwise, `SDIO_TIEH = RTC_CNTL_SDIO_TIEH`.

#### 29.3.4.4 Brownout Detector

The brownout detector checks the voltage of pins `VDD3P3_RTC_IO`, `VDD3P3_CPU`, `VDDA1`, and `VDDA2`. If the voltage of these pins drops rapidly and becomes too low, the detector would trigger a signal to shut down some power-consuming blocks (such as LNA, PA, etc.) to allow extra time for the digital system to save and transfer important data.

The brownout detector has ultra-low power consumption and remains enabled whenever the chip is powered up. For the architecture of the ESP32-S2 brownout detector, see Figure 29-9.

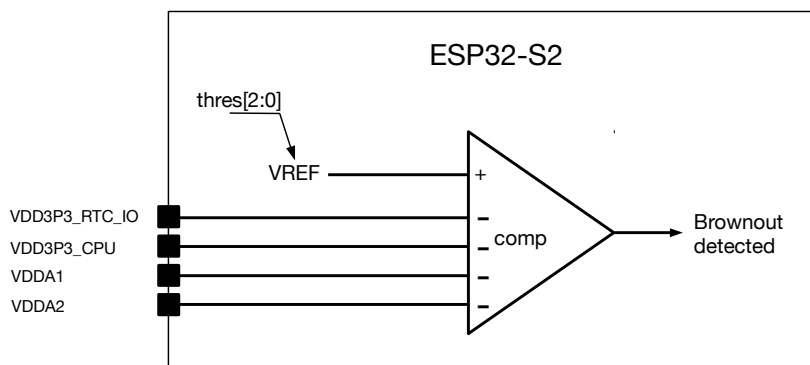


Figure 29-9. Brown-out detector

1. `RTC_CNTL_RTC_BROWN_OUT_DET` indicates the output level of brown-out detector. This register is low level by default, and outputs high level when the voltage of the detected pin drops below the predefined threshold;
2. I2C register `ULP_CAL_REG5[2:0]` configures the trigger threshold of the brown-out detector;

Table 166: Brown-out Detector Configuration

ULP_CAL_REG5[2:0]	VDD (Unit: V)
0	2.67
1	3.30
2	3.19
3	2.98
4	2.84
5	2.67
6	2.56
7	2.44

3. [RTC\\_CNTL\\_BROWN\\_OUT\\_RST\\_SEL](#) configures the reset type.

- 0: resets the chip
- 1: resets the system

**Note:**

For more information regarding chip reset and system reset, please refer to [2 Reset and Clock](#).

## 29.4 Power Modes Management

### 29.4.1 Power Domain

ESP32-S2 has 10 power domains in three power domain categories:

- RTC
  - Power management unit
  - RTC peripherals
  - RTC slow memory
  - RTC fast memory
- Digital
  - Digital core
  - Wi-Fi digital circuits
- Analog
  - 8 MHz crystals
  - 40 MHz crystals
  - PLL
  - RF circuits

### 29.4.2 RTC States

ESP32-S2 has three main RTC states: Active, Monitor, and Sleep. The transition process among these states can be seen in [Figure 29-10](#).

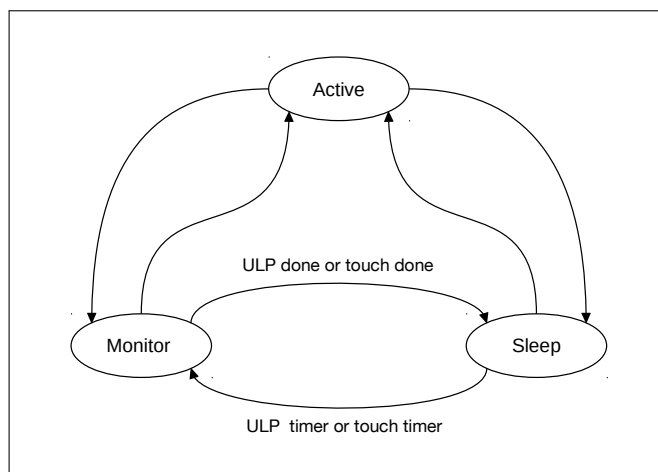


Figure 29-10. RTC States

Under different RTC states, different power domains are powered up or down by default, but can also be force-powered-up (FPU) or force-powered-down (FPD) individually based on actual requirements. For details, please refer to Table 167.

Table 167: RTC Statues Transition

Power Domain		RTC States			Notes
Category	Sub-category	Active	Monitor	Sleep	
RTC	Power Management Unit	ON	ON	ON	1
	RTC Peripherals	ON	ON	OFF	2
	RTC Slow Memory	ON	OFF	OFF	3
	RTC Fast Memory	ON	OFF	OFF	4
Digital	Digital Core	ON	OFF	OFF	5
	Wi-Fi Digital Circuits	ON	OFF	OFF	6
Analog	8 MHz Crystals	ON	ON	OFF	-
	40 MHz Crystals	ON	OFF	OFF	-
	PLL	ON	OFF	OFF	-
	RF Circuits	-	-	-	-

**Note:**

1. ESP32-S2's power management unit is specially designed to be "always-on", which means it is always on when the chip is powered up. Therefore, users cannot FPU or FPD the power management unit.
2. The RTC peripherals include [28 ULP Coprocessor](#) and on-chip sensor (i.e. temperature sensor controller, touch sensor, SAR ADC controller).
3. RTC slow memory supports 8 KB SRAM, which can be used to reserve memory or to store ULP instructions and / or data. This memory can be accessed by CPU via PeriBUS2 (starting address is 0x50000000), and should be force-power-on.
4. RTC fast memory supports 8 KB SRAM, which can be used to reserve memory. This memory can be accessed by CPU via IRAM0 / DRAM0, and should be forced-power-up.
5. When the digital core of the digital system is powered down, all components in the digital system are turned off.

It's worth noting that, ESP32-S2's ROM and SRAM are no longer controlled as independent power domains, thus cannot be force-powered-up or force-powered-down when the digital core is powered down.

6. Power domain Wi-Fi digital circuits includes Wi-Fi MAC and BB (Base Band).

### 29.4.3 Pre-defined Power Modes

As mentioned earlier, ESP32-S2 has five power modes, which are predefined configurations that power up different combinations of power domains. For details, please refer to Table 168.

**Table 168: Predefined Power Modes**

Power Modes	Power Domain									
	PMU	RTC Pe-ripher-als	RTC Slow Mem-ory	RTC Fast Mem-ory	Digital Core	Wi-Fi Digital Cir-cuits	8 MHz Crys-tals	40 MHz Crys-tals	PLL	RF Cir-cuits
Active	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON
Modem-sleep	ON	ON	ON	ON	ON	ON *	ON	ON	ON	OFF
Light-sleep	ON	ON	ON	ON	ON	ON *	OFF	OFF	OFF	OFF
Deep-sleep	ON	ON	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF
Hibernation	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

**Note:**

\* Configurable

By default, ESP32-S2 first enters the Active mode after system resets, then enters different low-power modes (including Modem-sleep, Light-sleep, Deep-sleep, and Hibernation) to save power after the CPU stalls for a specific time (For example, when CPU is waiting to be wakened up by an external event). From modes Active to Hibernation, the performance<sup>1</sup> and power consumption<sup>2</sup> decreases and wakeup latency increases. Also, the supported wakeup sources for different power modes are different<sup>3</sup>. Users can choose a power mode based on their requirements of performance, power consumption, wakeup latency, and available wakeup sources.

**Note:**

1. For details, please refer to Table 168.
2. For details on power consumption, please refer to the Current Consumption Characteristics in [ESP32-S2 Datasheet](#).
3. For details on the supported wakeup sources, please refer to Section 29.4.4.

### 29.4.4 Wakeup Source

The ESP32-S2 supports various wakeup sources, which could wake up the CPU in different sleep modes. The wakeup source is determined by `RTC_CNTL_RTC_WAKEUP_ENA` as shown in Table 169.

**Table 169: Wakeup Source**

WAKEUP_ENA	Wakeup Source	Light-sleep	Deep-sleep	Hibernation	Notes*
0x1	EXT0	Y	Y	-	<a href="#">1</a>
0x2	EXT1	Y	Y	Y	<a href="#">2</a>
0x4	GPIO	Y	Y	-	<a href="#">3</a>
0x8	RTC timer	Y	Y	Y	-
0x20	Wi-Fi	Y	-	-	<a href="#">4</a>
0x40	UART0	Y	-	-	<a href="#">5</a>
0x80	UART1	Y	-	-	<a href="#">5</a>
0x100	TOUCH	Y	Y	-	<a href="#">6</a>
0x800	ULP-FSM	Y	Y	-	<a href="#">7</a>
0x1000	XTAL_32K	Y	Y	Y	<a href="#">8</a>
0x2000	ULP-RISCV Trap	Y	Y	-	<a href="#">9</a>
0x8000	USB	Y	-	-	<a href="#">10</a>

**Note:**

- EXT0 can only wake up the chip from Light-sleep / Deep-sleep modes. If [RTC\\_CNTL\\_EXT\\_WAKEUP0\\_LV](#) is 1, it's triggered when the pin level is high. Otherwise, it's triggered when the pin level is low. Users can configure [RTCIO\\_EXT\\_WAKEUP0\\_SEL](#) to select an RTC pin as a wakeup source.
- EXT1 is especially designed to wake up the chip from any sleep modes, and can be triggered by a combination of pins. Users should define the combination of wakeup sources by configuring [RTC\\_CNTL\\_EXT\\_WAKEUP1\\_SEL\[17:0\]](#) according to the bitmap of selected wakeup source. When [RTC\\_CNTL\\_EXT\\_WAKEUP1\\_LV](#) == 1, the chip is waken up if any pin in the combination is high level. When [RTC\\_CNTL\\_EXT\\_WAKEUP1\\_LV](#) == 0, the chip is only waken up if all pins in the combination is low level.
- In Deep-sleep mode, only the RTC GPIOs (not regular GPIOs) can work as a wakeup source.
- To wake up the chip with a Wi-Fi source, the chip switches between the Active, Modem-sleep, and Light-sleep modes. The CPU and RF modules are woken up at predetermined intervals to keep Wi-Fi connections active.
- A wakeup is triggered when the number of RX pulses received exceeds the setting in the threshold register.
- A wakeup is triggered when any touch event is detected by the touch sensor.
- A wakeup is triggered when [RTC\\_CNTL\\_RTC\\_SW\\_CPU\\_INT](#) is configured by the ULP co-processor.
- When the 32 kHz crystal is working as RTC slow clock, a wakeup is triggered upon any detection of any crystal stops by the 32 kHz watchdog timer.
- A wakeup is triggered when the ULP co-processor starts capturing exceptions (e.g., stack overflow).
- A wakeup is triggered when the USB host sends USB into the RESUMING state.

## 29.5 RTC Boot

The wakeup time for Deep-sleep and Hibernation modes are much longer, compared to the Light-sleep and Modem-sleep, because the ROMs and RAMs are both powered down in this case, and the CPU needs more time for ROM unpacking and data-copying from the flash (SPI booting). However, it's worth noting that both RTC fast memory and RTC slow memory remain powered up in the Deep-sleep mode. Therefore, users can store codes (so called “deep sleep wake stub” of up to 8 KB) either in RTC fast memory or RTC slow memory to avoid the above-mentioned ROM unpacking and SPI booting, thus speeding up the wakeup process.

### Method one: Using RTC slow memory

1. Set [RTC\\_CNTL\\_PROCPU\\_STAT\\_VECTOR\\_SEL](#) to 0.
2. Send the chip into sleep.
3. After the CPU is powered up, the reset vector starts resetting from 0x50000000 instead of 0x40000400, which does not involve any ROM unpacking and SPI booting. The codes stored in RTC slow memory starts running immediately after the CPU reset. However, note that the content in the RTC slow memory should be initialized before sending the chip into sleep.

### Method two: Using RTC fast memory

1. Set [RTC\\_CNTL\\_PROCPU\\_STAT\\_VECTOR\\_SEL](#) to 1.
2. Calculate CRC for the RTC fast memory, and save the result in [RTC\\_CNTL\\_STORE6\\_REG\[31:0\]](#).
3. Set [RTC\\_CNTL\\_STORE7\\_REG\[31:0\]](#) to the entry address of RTC fast memory.
4. Send the chip into sleep.
5. ROM unpacking and some of the initialization starts after the CPU is powered up. After that, calculate the CRC for the RTC fast memory again. If the result matches with register [RTC\\_CNTL\\_STORE6\\_REG\[31:0\]](#), the CPU jumps to the entry address.

The boot flow is shown in Figure [29-11](#).

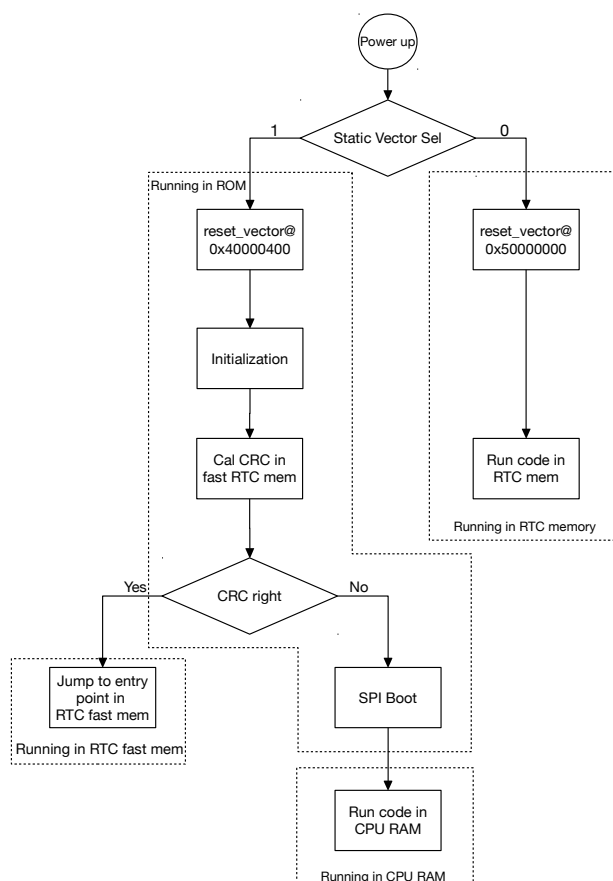


Figure 29-11. ESP32-S2 Boot Flow

When working under low-power modes, ESP32-S2's 40 MHz crystal oscillator and PLL are usually powered down to reduce power consumption. However, the low-power clock remains on so the chip can operate properly under low-power modes.

## 29.6 Base Address

Users can access the low-power management module of ESP32-S2 with two base addresses, which can be seen in Table 170. For more information about accessing peripherals from different buses please see Chapter 1 *System and Memory*.

Table 170: Low-power Management Base Address

Bus to Access Peripheral	Base Address
PeriBUS1	0x3f408000
PeriBUS2	0x60008000

## 29.7 Register Summary

The addresses in the following table are relative to the Low Power Management base addresses provided in Section 29.6.

Name	Description	Address	Access
<b>RTC Option Registers</b>			
<a href="#">RTC_CNTL_OPTIONS0_REG</a>	Sets the power options of crystal and PLL clocks, and initiates reset by software	0x0000	varies
<a href="#">RTC_CNTL_OPTION1_REG</a>	RTC option register	0x0128	R/W
<b>RTC Timer Registers</b>			
<a href="#">RTC_CNTL_SLP_TIMER0_REG</a>	RTC timer threshold register 0	0x0004	R/W
<a href="#">RTC_CNTL_SLP_TIMER1_REG</a>	RTC timer threshold register 1	0x0008	varies
<a href="#">RTC_CNTL_TIME_UPDATE_REG</a>	RTC timer update control register	0x000C	varies
<a href="#">RTC_CNTL_TIME_LOW0_REG</a>	Stores the lower 32 bits of RTC timer 0	0x0010	RO
<a href="#">RTC_CNTL_TIME_HIGH0_REG</a>	Stores the higher 16 bits of RTC timer 0	0x0014	RO
<a href="#">RTC_CNTL_STATE0_REG</a>	Configures the sleep / reject / wakeup state	0x0018	varies
<a href="#">RTC_CNTL_TIMER1_REG</a>	Configures CPU stall options	0x001C	R/W
<a href="#">RTC_CNTL_TIMER2_REG</a>	Configures RTC slow clock and touch controller	0x0020	R/W
<a href="#">RTC_CNTL_TIMER5_REG</a>	Configures the minimal sleep cycles	0x002C	R/W
<a href="#">RTC_CNTL_TIME_LOW1_REG</a>	Stores the lower 32 bits of RTC timer 1	0x00E8	RO
<a href="#">RTC_CNTL_TIME_HIGH1_REG</a>	Stores the higher 16 bits of RTC timer 1	0x00EC	RO
<b>Internal Power Control Register</b>			
<a href="#">RTC_CNTL_ANA_CONF_REG</a>	Configures the power options for I2C and PLLA	0x0034	R/W
<a href="#">RTC_CNTL_REG</a>	Low-power voltage and digital voltage regulators configuration register	0x0084	R/W
<a href="#">RTC_CNTL_PWC_REG</a>	RTC power configuration register	0x0088	R/W
<a href="#">RTC_CNTL_DIG_PWC_REG</a>	Digital system power configuration register	0x008C	R/W
<a href="#">RTC_CNTL_DIG_ISO_REG</a>	Digital system isolation configuration register	0x0090	varies
<a href="#">RTC_CNTL_LOW_POWER_ST_REG</a>	RTC main state machine state register	0x00CC	RO
<b>Reset Control Register</b>			
<a href="#">RTC_CNTL_RESET_STATE_REG</a>	Indicates the CPU reset source	0x0038	varies
<b>Sleep and Wake-up Control Register</b>			
<a href="#">RTC_CNTL_WAKEUP_STATE_REG</a>	Wakeup bitmap enabling register	0x003C	R/W
<a href="#">RTC_CNTL_EXT_WAKEUP_CONF_REG</a>	GPIO wakeup configuration register	0x0064	R/W
<a href="#">RTC_CNTL_SLP_REJECT_CONF_REG</a>	Configures sleep / reject options	0x0068	R/W
<a href="#">RTC_CNTL_EXT_WAKEUP1_REG</a>	EXT1 wakeup configuration register	0x00DC	varies
<a href="#">RTC_CNTL_EXT_WAKEUP1_STATUS_REG</a>	EXT1 wakeup source register	0x00E0	RO
<a href="#">RTC_CNTL_SLP_REJECT_CAUSE_REG</a>	Stores the reject-to-sleep cause	0x0124	RO
<a href="#">RTC_CNTL_SLP_WAKEUP_CAUSE_REG</a>	Stores the sleep-to-wakeup cause	0x012C	RO
<b>Interrupt Registers</b>			
<a href="#">RTC_CNTL_INT_ENA_RTC_REG</a>	RTC interrupt enabling register	0x0040	R/W
<a href="#">RTC_CNTL_INT_RAW_RTC_REG</a>	RTC interrupt raw register	0x0044	RO
<a href="#">RTC_CNTL_INT_ST_RTC_REG</a>	RTC interrupt state register	0x0048	RO
<a href="#">RTC_CNTL_INT_CLR_RTC_REG</a>	RTC interrupt clear register	0x004C	WO
<b>Reservation Registers</b>			
<a href="#">RTC_CNTL_STORE0_REG</a>	Reservation register 0	0x0050	R/W



Name	Description	Address	Access
<a href="#">RTC_CNTL_STORE1_REG</a>	Reservation register 1	0x0054	R/W
<a href="#">RTC_CNTL_STORE2_REG</a>	Reservation register 2	0x0058	R/W
<a href="#">RTC_CNTL_STORE3_REG</a>	Reservation register 3	0x005C	R/W
<a href="#">RTC_CNTL_STORE4_REG</a>	Reservation register 4	0x00BC	R/W
<a href="#">RTC_CNTL_STORE5_REG</a>	Reservation register 5	0x00C0	R/W
<a href="#">RTC_CNTL_STORE6_REG</a>	Reservation register 6	0x00C4	R/W
<a href="#">RTC_CNTL_STORE7_REG</a>	Reservation register 7	0x00C8	R/W
<b>Clock Control Registers</b>			
<a href="#">RTC_CNTL_EXT_XTL_CONF_REG</a>	32 kHz crystal oscillator configuration register	0x0060	varies
<a href="#">RTC_CNTL_CLK_CONF_REG</a>	RTC clock configuration register	0x0074	R/W
<a href="#">RTC_CNTL_SLOW_CLK_CONF_REG</a>	RTC slow clock configuration register	0x0078	R/W
<a href="#">RTC_CNTL_XTAL32K_CLK_FACTOR_REG</a>	Configures the divider for the backup clock of 32 kHz crystal oscillator	0x00F0	R/W
<a href="#">RTC_CNTL_XTAL32K_CONF_REG</a>	32 kHz crystal oscillator configuration register	0x00F4	R/W
<b>RTC Watchdog Control Register</b>			
<a href="#">RTC_CNTL_WDTCONFIG0_REG</a>	RTC watchdog configuration register	0x0094	R/W
<a href="#">RTC_CNTL_WDTCONFIG1_REG</a>	Configures the hold time of RTC watchdog at level 1	0x0098	R/W
<a href="#">RTC_CNTL_WDTCONFIG2_REG</a>	Configures the hold time of RTC watchdog at level 2	0x009C	R/W
<a href="#">RTC_CNTL_WDTCONFIG3_REG</a>	Configures the hold time of RTC watchdog at level 3	0x00A0	R/W
<a href="#">RTC_CNTL_WDTCONFIG4_REG</a>	Configures the hold time of RTC watchdog at level 4	0x00A4	R/W
<a href="#">RTC_CNTL_WDTFEED_REG</a>	RTC watchdog SW feed configuration register	0x00A8	WO
<a href="#">RTC_CNTL_WDTWPROTECT_REG</a>	RTC watchdog write protection configuration register	0x00AC	R/W
<a href="#">RTC_CNTL_SWD_CONF_REG</a>	Super watchdog configuration register	0x00B0	varies
<a href="#">RTC_CNTL_SWD_WPROTECT_REG</a>	Super watchdog write protection configuration register	0x00B4	R/W
<b>Other Registers</b>			
<a href="#">RTC_CNTL_SW_CPU_STALL_REG</a>	CPU stall configuration register	0x00B8	R/W
<a href="#">RTC_CNTL_PAD_HOLD_REG</a>	Configures the hold options for RTC GPIOs	0x00D4	R/W
<a href="#">RTC_CNTL_DIG_PAD_HOLD_REG</a>	Configures the hold options for digital GPIOs	0x00D8	R/W
<a href="#">RTC_CNTL_BROWN_OUT_REG</a>	Brownout configuration register	0x00E4	varies

## 29.8 Registers

The addresses in this section are relative to the Low Power Management base addresses provided in Section 29.6.

Register 29.1: RTC\_CNTL\_OPTIONS0\_REG (0x0000)

RTC_CNTL_SW_SYS_RST RTC_CNTL_DG_WRAP_FORCE_NORST RTC_CNTL_DG_WRAP_FORCE_RST  (reserved)																															RTC_CNTL_XTL_FORCE_PU RTC_CNTL_XTL_FORCE_PD RTC_CNTL_BBPLL_FORCE_PU RTC_CNTL_BBPLL_FORCE_PD RTC_CNTL_BBPLL_I2C_FORCE_PU RTC_CNTL_BBPLL_I2C_FORCE_PD RTC_CNTL_BB_I2C_FORCE_PU RTC_CNTL_BB_I2C_FORCE_PD (reserved) RTC_CNTL_SW_PROCPU_RST (reserved) RTC_CNTL_SW_STALL_PROCPU_C0															
31	30	29	28													14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset																		

**RTC\_CNTL\_SW\_STALL\_PROCPU\_C0** When [RTC\\_CNTL\\_SW\\_STALL\\_PROCPU\\_C1](#) is configured to 0x21, setting this bit to 0x2 stalls the CPU by SW.

**RTC\_CNTL\_SW\_PROCPU\_RST** Set this bit to reset the CPU by SW. (WO)

**RTC\_CNTL\_BB\_I2C\_FORCE\_PD** Set this bit to FPD BB\_I2C. (R/W)

**RTC\_CNTL\_BB\_I2C\_FORCE\_PU** Set this bit to FPU BB\_I2C. (R/W)

**RTC\_CNTL\_BBPLL\_I2C\_FORCE\_PD** Set this bit to FPD BB\_PLL\_I2C. (R/W)

**RTC\_CNTL\_BBPLL\_I2C\_FORCE\_PU** Set this bit to FPU BB\_PLL\_I2C. (R/W)

**RTC\_CNTL\_BBPLL\_FORCE\_PD** Set this bit to FPD BB\_PLL. (R/W)

**RTC\_CNTL\_BBPLL\_FORCE\_PU** Set this bit to FPU BB\_PLL. (R/W)

**RTC\_CNTL\_XTL\_FORCE\_PD** Set this bit to FPD the crystal oscillator. (R/W)

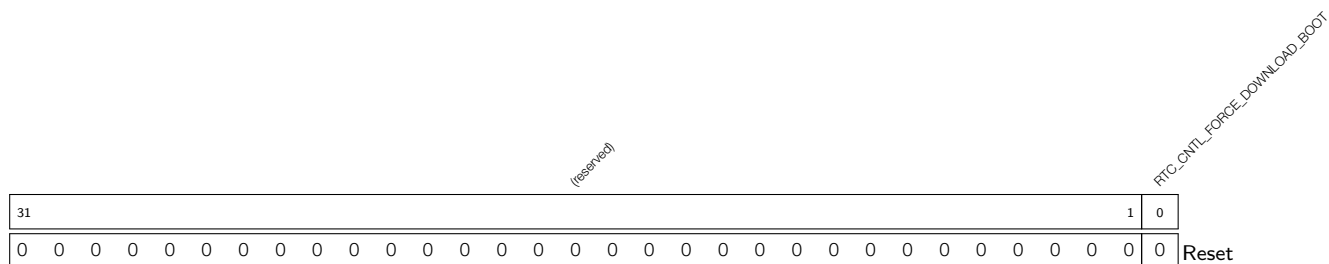
**RTC\_CNTL\_XTL\_FORCE\_PU** Set this bit to FPU the crystal oscillator. (R/W)

**RTC\_CNTL\_DG\_WRAP\_FORCE\_RST** Set this bit to force reset the digital system in deep-sleep. (R/W)

**RTC\_CNTL\_DG\_WRAP\_FORCE\_NORST** Set this bit to disable force reset to digital system in deep-sleep. (R/W)

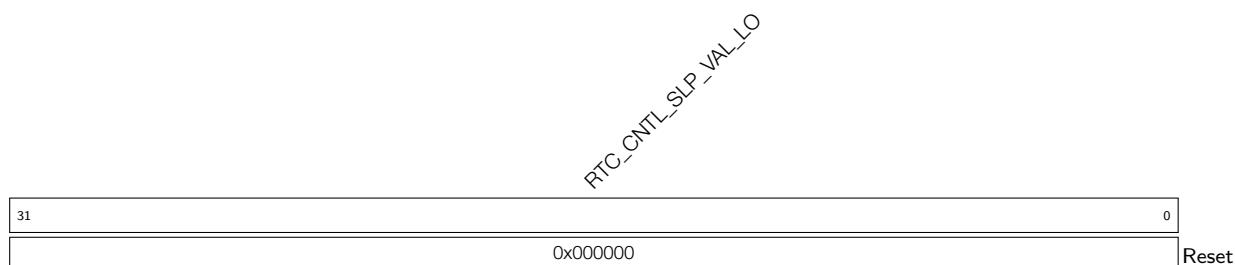
**RTC\_CNTL\_SW\_SYS\_RST** Set this bit to reset the system via SW. (WO)

Register 29.2: RTC\_CNTL\_OPTION1\_REG (0x0128)



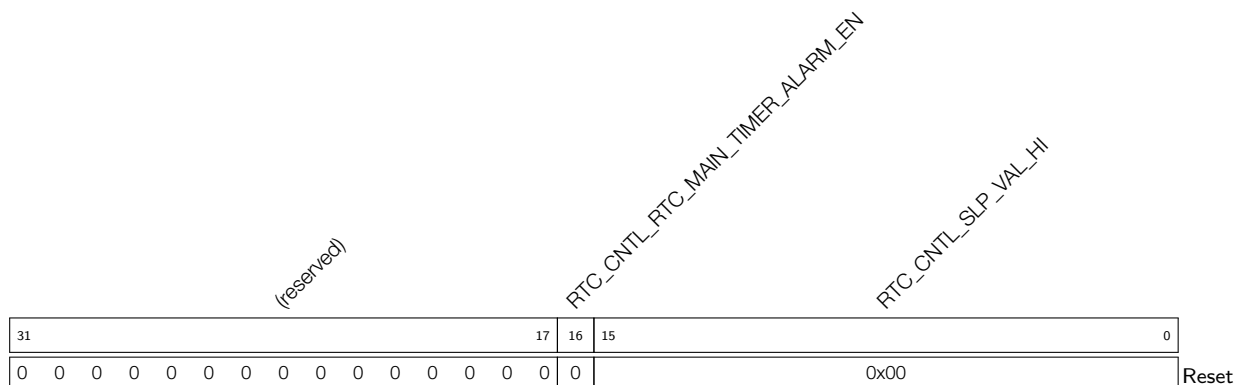
**RTC\_CNTL\_FORCE\_DOWNLOAD\_BOOT** Set this bit to force the chip to boot from the download mode. (R/W)

Register 29.3: RTC\_CNTL\_SLP\_TIMER0\_REG (0x0004)



**RTC\_CNTL\_SLP\_VAL\_LO** Sets the lower 32 bits of the trigger threshold for the RTC timer. (R/W)

Register 29.4: RTC\_CNTL\_SLP\_TIMER1\_REG (0x0008)

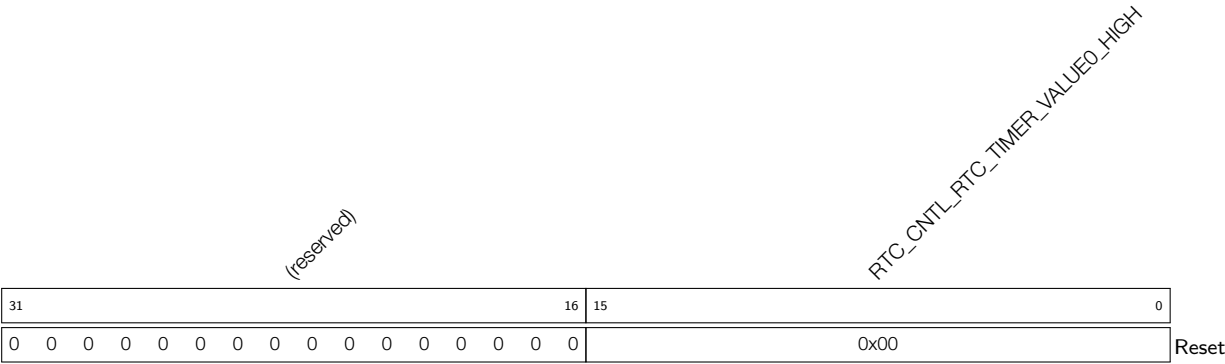


**RTC\_CNTL\_SLP\_VAL\_HI** Sets the higher 16 bits of the trigger threshold for the RTC timer. (R/W)

**RTC\_CNTL\_RTC\_MAIN\_TIMER\_ALARM\_EN** Sets this bit to enable the timer alarm. (WO)

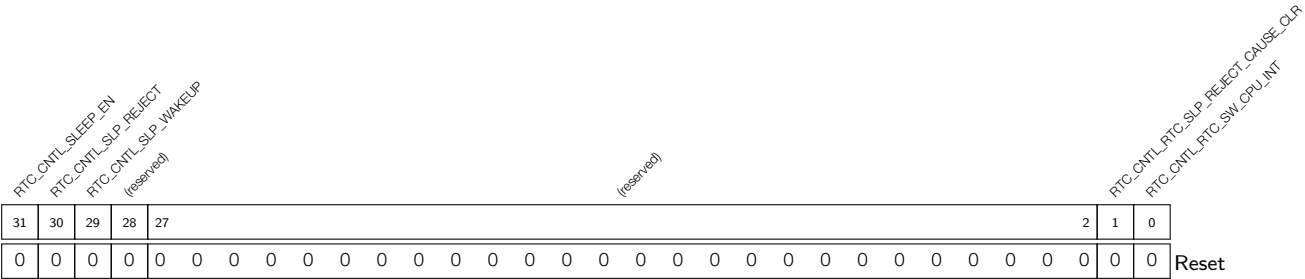


Register 29.7: RTC\_CNTL\_TIME\_HIGH0\_REG (0x0014)



**RTC\_CNTL\_RTC\_TIMER\_VALUE0\_HIGH** Stores the higher 16 bits of RTC timer 0. (RO)

Register 29.8: RTC\_CNTL\_STATE0\_REG (0x0018)



**RTC\_CNTL\_RTC\_SW\_CPU\_INT** Sends a SW RTC interrupt to CPU. (WO)

**RTC\_CNTL\_RTC\_SLP\_REJECT\_CAUSE\_CLR** Clears the RTC reject-to-sleep cause. (WO)

**RTC\_CNTL\_SLP\_WAKEUP** Sleep wakeup bit. (R/W)

**RTC\_CNTL\_SLP\_REJECT** Sleep reject bit. (R/W)

**RTC\_CNTL\_SLEEP\_EN** Sends the chip to sleep. (R/W)

**Register 29.9: RTC\_CNTL\_TIMER1\_REG (0x001C)**

RTC_CNTL_PLL_BUF_WAIT									
RTC_CNTL_XTL_BUF_WAIT									
RTC_CNTL_OX8M_WAIT									
RTC_CNTL_CPU_STALL_WAIT									
RTC_CNTL_CPU_STALL_EN									
31	24	23	14	13	6	5	1	0	
40		80			0x10		1	1	Reset

**RTC\_CNTL\_CPU\_STALL\_EN** Enables the CPU stalling. (R/W)

**RTC\_CNTL\_CPU\_STALL\_WAIT** Sets the CPU stall waiting cycles (using the RTC fast clock). (R/W)

**RTC\_CNTL\_CK8M\_WAIT** Sets the 8 MHz clock waiting cycles (using the RTC slow clock). (R/W)

**RTC\_CNTL\_XTL\_BUF\_WAIT** Sets the XTAL waiting cycles (using the RTC slow clock). (R/W)

**RTC\_CNTL\_PLL\_BUF\_WAIT** Sets the PLL waiting cycles (using the RTC slow clock). (R/W)

**Register 29.10: RTC\_CNTL\_TIMER2\_REG (0x0020)**

RTC_CNTL_MIN_TIME_CK8M_OFF																RTC_CNTL_ULPCP_TOUCH_START_WAIT																(reserved)																							
31								24								23								15								14								0															
0x1																0x10																0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0								0								Reset							

**RTC\_CNTL\_ULPCP\_TOUCH\_START\_WAIT** Sets the waiting cycles (using the RTC slow clock) before the ULP co-processor or touch controller starts to work. (R/W)

**RTC\_CNTL\_MIN\_TIME\_CK8M\_OFF** Sets the minimal cycles for 8 MHz clock (using the RTC slow clock) when powered down. (R/W)

**Register 29.11: RTC\_CNTL\_TIMER5\_REG (0x002C)**

(reserved)																RTC_CNTL_MIN_SLP_VAL								(reserved)																																															
31																16																15																8								7								0							
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x80																0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																Reset																							

**RTC\_CNTL\_MIN\_SLP\_VAL** Sets the minimal sleep cycles (using the RTC slow clock). (R/W)

**Register 29.12: RTC\_CNTL\_TIME\_LOW1\_REG (0x00E8)**

RTC_CNTL_RTC_TIMER_VALUE1_LOW																																
31																															0	
0x000000																																Reset

**RTC\_CNTL\_RTC\_TIMER\_VALUE1\_LOW** Stores the lower 32 bits of RTC timer 1. (RO)

**Register 29.13: RTC\_CNTL\_TIME\_HIGH1\_REG (0x00EC)**

(reserved)																RTC_CNTL_RTC_TIMER_VALUE1_HIGH																															
31																15																0															
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x00																Reset															

**RTC\_CNTL\_RTC\_TIMER\_VALUE1\_HIGH** Stores the higher 16 bits of RTC timer. (RO)

Register 29.14: RTC\_CNTL\_ANA\_CONF\_REG (0x0034)

(reserved)								RTC_CNTL_PLLA_FORCE_PU		RTC_CNTL_PLLA_FORCE_PD		RTC_CNTL_SAR_I2C_FORCE_PU		RTC_CNTL_SAR_I2C_FORCE_PD		RTC_CNTL_GLITCH_RST_EN		(reserved)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
31								25								24	23	22	21	20	19																			0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**RTC\_CNTL\_GLITCH\_RST\_EN** Set this bit to enable a reset when the system detects a glitch. (R/W)

**RTC\_CNTL\_SAR\_I2C\_FORCE\_PD** Set this bit to FPD the SAR\_I2C. (R/W)

**RTC\_CNTL\_SAR\_I2C\_FORCE\_PU** Set this bit to FPU the SAR\_I2C. (R/W)

**RTC\_CNTL\_PLLA\_FORCE\_PD** Set this bit to FPD the PLLA. (R/W)

**RTC\_CNTL\_PLLA\_FORCE\_PU** Sets this bit to FPU the PLLA. (R/W)



Register 29.15: RTC\_CNTL\_REG (0x0084)

RTC_CNTL_RTC_REGULATOR_FORCE_PU		RTC_CNTL_RTC_REGULATOR_FORCE_PD		(reserved)		RTC_CNTL_RTC_DBIAS_WAK		RTC_CNTL_RTC_DBIAS_SLP		RTC_CNTL_SCK_DCAP		RTC_CNTL_DIG_DBIAS_WAK		RTC_CNTL_DIG_DBIAS_SLP		(reserved)	
31	30	29	28	27	25	24	22	21	14	13	11	10	8	7			0
1	0	0	0		4		4		0		4		4		0	0	0

Reset

**RTC\_CNTL\_DIG\_DBIAS\_SLP** Configures the regulation factor for the digital system voltage regulator when the CPU is in sleep state. (R/W)

**RTC\_CNTL\_DIG\_DBIAS\_WAK** Configures the regulation factor for the digital system voltage regulator when the CPU is in active status. (R/W)

**RTC\_CNTL\_SCK\_DCAP** Configures the frequency of the RTC clocks. (R/W)

**RTC\_CNTL\_RTC\_DBIAS\_SLP** Configures the regulation factor for the low-power voltage regulator when the CPU is in sleep status. (R/W)

**RTC\_CNTL\_RTC\_DBIAS\_WAK** Configures the regulation factor for the low-power voltage regulator when the CPU is in active status. (R/W)

**RTC\_CNTL\_RTC\_REGULATOR\_FORCE\_PD** Set this bit to FPD the low-power voltage regulator, which means decreasing its voltage to 0.8 V or lower. (R/W)

**RTC\_CNTL\_RTC\_REGULATOR\_FORCE\_PU** Set this bit to FPU the low-power voltage regulator, which means increasing its voltage to higher than 0.8 V. (R/W)

**Register 29.16: RTC\_CNTL\_PWC\_REG (0x0088)**

(reserved)										RTC_CNTL_RTC_PWD_FORCE_HOLD RTC_CNTL_RTC_PD_EN RTC_CNTL_RTC_FORCE_PU RTC_CNTL_RTC_FORCE_PD RTC_CNTL_RTC_SLOWMEM_PD_EN RTC_CNTL_RTC_SLOWMEM_FORCE_PU RTC_CNTL_RTC_SLOWMEM_FORCE_PD RTC_CNTL_RTC_FASTMEM_PD_EN RTC_CNTL_RTC_FASTMEM_FORCE_PU RTC_CNTL_RTC_SLOWMEM_FORCE_PD RTC_CNTL_RTC_SLOWMEM_FORCE_LPU RTC_CNTL_RTC_FASTMEM_FORCE_LPU RTC_CNTL_RTC_FASTMEM_FORCE_LPD RTC_CNTL_RTC_FASTMEM_FORCE_LPU RTC_CNTL_RTC_FORCE_NOISO RTC_CNTL_RTC_SLOWMEM_FORCE_ISO RTC_CNTL_RTC_SLOWMEM_FORCE_NOISO RTC_CNTL_RTC_FASTMEM_FORCE_NOISO																					
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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	0	1	0	1
Reset																															

Reset

**RTC\_CNTL\_RTC\_FASTMEM\_FORCE\_NOISO** Set this bit to disable the force isolation of the RTC fast memory. (R/W)

**RTC\_CNTL\_RTC\_FASTMEM\_FORCE\_ISO** Set this bit to force isolation of the RTC fast memory. (R/W)

**RTC\_CNTL\_RTC\_SLOWMEM\_FORCE\_NOISO** Set this bit to disable the force isolation of the RTC slow memory. (R/W)

**RTC\_CNTL\_RTC\_SLOWMEM\_FORCE\_ISO** Set this bit to force isolation of the RTC slow memory. (R/W)

**RTC\_CNTL\_RTC\_FORCE\_ISO** Set this bit to force isolation of the RTC peripherals. (R/W)

**RTC\_CNTL\_RTC\_FORCE\_NOISO** Set this bit to disable the force isolation of the RTC peripherals. (R/W)

**RTC\_CNTL\_RTC\_FASTMEM\_FOLW\_CPU** Set this bit to FPD the RTC fast memory when the CPU is powered down. Reset this bit to FPD the RTC fast memory when the RTC main state machine is powered down. (R/W)

**RTC\_CNTL\_RTC\_FASTMEM\_FORCE\_LPD** Set this bit to force not retain the RTC fast memory. (R/W)

**RTC\_CNTL\_RTC\_FASTMEM\_FORCE\_LPU** Set this bit to force retain the RTC fast memory. (R/W)

**RTC\_CNTL\_RTC\_SLOWMEM\_FOLW\_CPU** Set this bit to FPD the RTC slow memory when the CPU is powered down. Reset this bit to FPD the RTC slow memory when the RTC main state machine is powered down. (R/W)

**RTC\_CNTL\_RTC\_SLOWMEM\_FORCE\_LPD** Set this bit to force not retain the RTC slow memory. (R/W)

**RTC\_CNTL\_RTC\_SLOWMEM\_FORCE\_LPU** Set this bit to force retain the RTC slow memory. (R/W)

**RTC\_CNTL\_RTC\_FASTMEM\_FORCE\_PD** Set this bit to FPD the RTC fast memory. (R/W)

**RTC\_CNTL\_RTC\_FASTMEM\_FORCE\_PU** Set this bit to FPU the RTC fast memory. (R/W)

**RTC\_CNTL\_RTC\_FASTMEM\_PD\_EN** Set this bit to enable PD for the RTC fast memory in sleep. (R/W)

Continued on the next page...

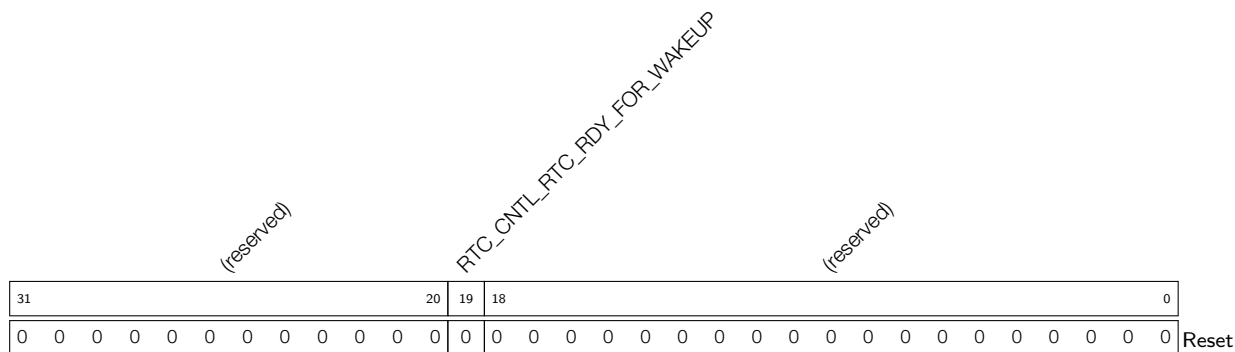


## 676

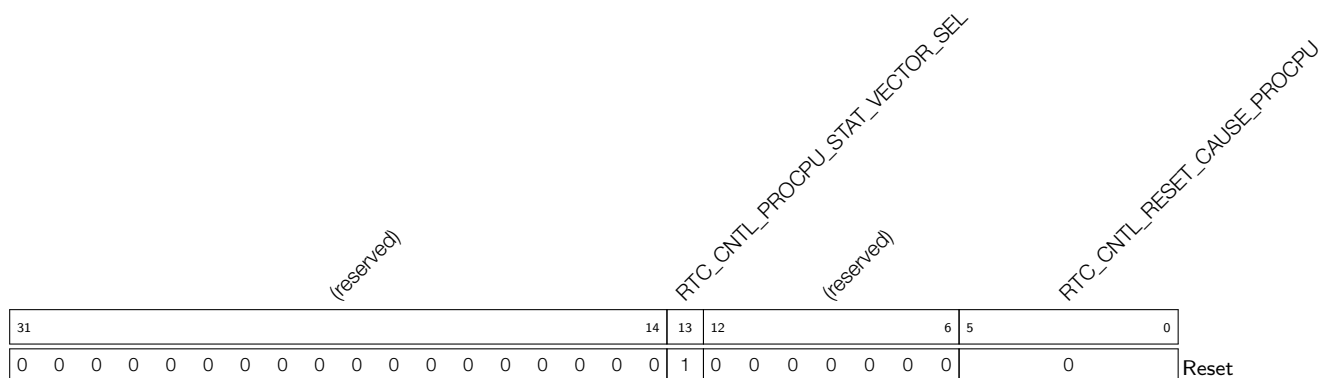
ESP32-S2 TRM (Preliminary V0.5)

[Submit Documentation Feedback](#)

**RTC\_CNTL\_DG\_WRAP\_FORCE\_NOISO** Set this bit to disable the force isolation of the digital system. (R/W)

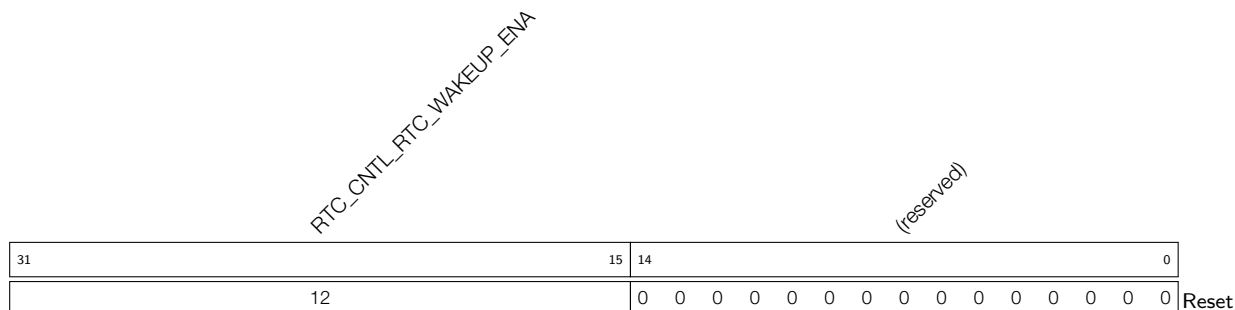
**Register 29.19: RTC\_CNTL\_LOW\_POWER\_ST\_REG (0x00CC)**

**RTC\_CNTL\_RTC\_RDY\_FOR\_WAKEUP** Indicates the RTC is ready to be triggered by any wakeup source. (RO)

**Register 29.20: RTC\_CNTL\_RESET\_STATE\_REG (0x0038)**

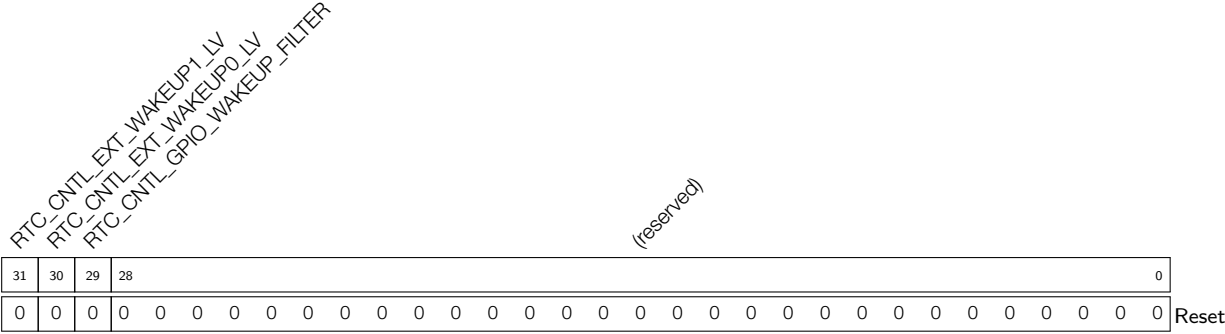
**RTC\_CNTL\_RESET\_CAUSE\_PROCPU** Stores the CPU reset cause. (RO)

**RTC\_CNTL\_PROCPU\_STAT\_VECTOR\_SEL** Selects the CPU static vector. (R/W)

**Register 29.21: RTC\_CNTL\_WAKEUP\_STATE\_REG (0x003C)**

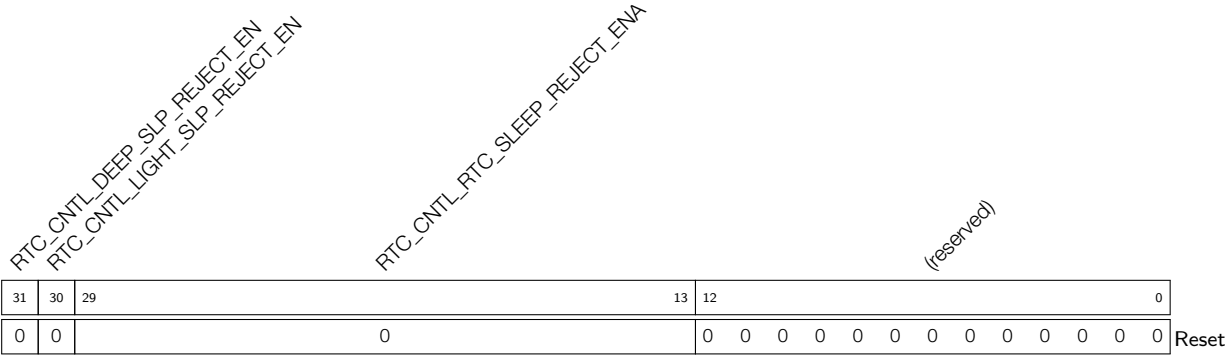
**RTC\_CNTL\_RTC\_WAKEUP\_ENA** Selects the wakeup source. For details, please refer to Table 169. (R/W)

Register 29.22: RTC\_CNTL\_EXT\_WAKEUP\_CONF\_REG (0x0064)



- RTC\_CNTL\_GPIO\_WAKEUP\_FILTER** Set this bit to enable the GPIO wakeup event filter. (R/W)
- RTC\_CNTL\_EXT\_WAKEUP0\_LV** 0: EXT0 at low level, 1: EXT0 at high level. (R/W)
- RTC\_CNTL\_EXT\_WAKEUP1\_LV** 0: EXT1 at low level, 1: EXT1 at high level. (R/W)

Register 29.23: RTC\_CNTL\_SLP\_REJECT\_CONF\_REG (0x0068)



- RTC\_CNTL\_RTC\_SLEEP\_REJECT\_ENA** Set this bit to enable reject-to-sleep. (R/W)
- RTC\_CNTL\_LIGHT\_SLP\_REJECT\_EN** Set this bit to enable reject-to-light-sleep. (R/W)
- RTC\_CNTL\_DEEP\_SLP\_REJECT\_EN** Set this bit to enable reject-to-deep-sleep. (R/W)

Register 29.24: RTC\_CNTL\_EXT\_WAKEUP1\_REG (0x00DC)

(reserved)										RTC_CNTL_EXT_WAKEUP1_STATUS_CLR										RTC_CNTL_EXT_WAKEUP1_SEL									
31											23	22	21													0			
0	0	0	0	0	0	0	0	0	0	0	0	0												Reset					

**RTC\_CNTL\_EXT\_WAKEUP1\_SEL** Selects a RTC GPIO to be the EXT1 wakeup source. (R/W)

**RTC\_CNTL\_EXT\_WAKEUP1\_STATUS\_CLR** Clears the EXT1 wakeup status. (WO)

Register 29.25: RTC\_CNTL\_EXT\_WAKEUP1\_STATUS\_REG (0x00E0)

(reserved)										RTC_CNTL_EXT_WAKEUP1_STATUS																															
31										22										21																					0
0 0 0 0 0 0 0 0 0 0										0																					Reset										

**RTC\_CNTL\_EXT\_WAKEUP1\_STATUS** Indicates the EXT1 wakeup status. (RO)

Register 29.26: RTC\_CNTL\_SLP\_REJECT\_CAUSE\_REG (0x0124)

(reserved)																RTC_CNTL_REJECT_CAUSE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																17																16																0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0															

**RTC\_CNTL\_REJECT\_CAUSE** Stores the reject-to-sleep cause. (RO)

**Register 29.27: RTC\_CNTL\_SLP\_WAKEUP\_CAUSE\_REG (0x012C)**

(reserved)																RTC_CNTL_WAKEUP_CAUSE																																															
31																17																16																0															
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0																Reset																															

**RTC\_CNTL\_WAKEUP\_CAUSE** Stores the wakeup cause. (RO)

**Register 29.28: RTC\_CNTL\_INT\_ENA\_RTC\_REG (0x0040)**

(reserved)																				RTC_CNTL_RTC_GLITCH_DET_INT_ENA											
																				RTC_CNTL_RTC_TOUCH_TIMEOUT_INT_ENA											
																				RTC_CNTL_RTC_TOUCH_TRAP_INT_ENA											
																				RTC_CNTL_RTC_XTAL32K_DEAD_INT_ENA											
																				RTC_CNTL_RTC_SWD_INT_ENA											
																				RTC_CNTL_RTC_SARADC2_INT_ENA											
																				RTC_CNTL_RTC_COCPU_INT_ENA											
																				RTC_CNTL_RTC_TSENS_INT_ENA											
																				RTC_CNTL_RTC_SARADC1_INT_ENA											
																				RTC_CNTL_RTC_MAIN_TIMER_INT_ENA											
																				RTC_CNTL_RTC_TOUCH_OUT_INT_ENA											
																				RTC_CNTL_RTC_TOUCH_INACTIVE_INT_ENA											
																				RTC_CNTL_RTC_TOUCH_ACTIVE_INT_ENA											
																				(reserved)											
																				RTC_CNTL_RTC_TOUCH_DONE_INT_ENA											
																				RTC_CNTL_RTC_TOUCH_SCAN_INT_ENA											
																				RTC_CNTL_RTC_WDT_INT_ENA											
																				RTC_CNTL_SLP_REJECT_INT_ENA											
																				RTC_CNTL_SLP_WAKEUP_INT_ENA											

31												20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset		

**RTC\_CNTL\_SLP\_WAKEUP\_INT\_ENA** Enables interrupts when the chip wakes up from sleep. (R/W)

**RTC\_CNTL\_SLP\_REJECT\_INT\_ENA** Enables interrupts when the chip rejects to go to sleep. (R/W)

**RTC\_CNTL\_RTC\_WDT\_INT\_ENA** Enables the RTC watchdog interrupt. (R/W)

**RTC\_CNTL\_RTC\_TOUCH\_SCAN\_DONE\_INT\_ENA** Enables interrupts upon the completion of a touch scanning. (R/W)

**RTC\_CNTL\_RTC\_ULP\_CP\_INT\_ENA** Enables the ULP co-processor interrupt. (R/W)

Continued on the next page...



**Register 29.28: RTC\_CNTL\_INT\_ENA\_RTC\_REG (0x0040)**

Continued from the previous page...

**RTC\_CNTL\_RTC\_TOUCH\_DONE\_INT\_ENA** Enables interrupts upon the completion of a single touch. (R/W)

**RTC\_CNTL\_RTC\_TOUCH\_ACTIVE\_INT\_ENA** Enables interrupts when a touch is detected. (R/W)

**RTC\_CNTL\_RTC\_TOUCH\_INACTIVE\_INT\_ENA** Enables interrupts when a touch is released. (R/W)

**RTC\_CNTL\_RTC\_BROWN\_OUT\_INT\_ENA** Enables the brownout interrupt. (R/W)

**RTC\_CNTL\_RTC\_MAIN\_TIMER\_INT\_ENA** Enables the RTC main timer interrupt. (R/W)

**RTC\_CNTL\_RTC\_SARADC1\_INT\_ENA** Enables the SAR ADC 1 interrupt. (R/W)

**RTC\_CNTL\_RTC\_TSENS\_INT\_ENA** Enables the temperature sensor interrupt. (R/W)

**RTC\_CNTL\_RTC\_COCPU\_INT\_ENA** Enables the ULP-RISCV interrupt. (R/W)

**RTC\_CNTL\_RTC\_SARADC2\_INT\_ENA** Enables the SAR ADC 2 interrupt. (R/W)

**RTC\_CNTL\_RTC\_SWD\_INT\_ENA** Enables the super watchdog interrupt. (R/W)

**RTC\_CNTL\_RTC\_XTAL32K\_DEAD\_INT\_ENA** Enables interrupts when the 32 kHz crystal is dead. (R/W)

**RTC\_CNTL\_RTC\_COCPU\_TRAP\_INT\_ENA** Enables interrupts when the ULP-RISCV is trapped. (R/W)

**RTC\_CNTL\_RTC\_TOUCH\_TIMEOUT\_INT\_ENA** Enables interrupts when touch sensor times out. (R/W)

**RTC\_CNTL\_RTC\_GLITCH\_DET\_INT\_ENA** Enables interrupts when a glitch is detected. (R/W)

Register 29.29: RTC\_CNTL\_INT\_RAW\_RTC\_REG (0x0044)

(reserved)																															RTC_CNTL_RTC_GLITCH_DET_INT_RAW RTC_CNTL_RTC_TOUCH_TIMEOUT_INT_RAW RTC_CNTL_RTC_COCPU_TRAP_INT_RAW RTC_CNTL_RTC_XTAL32K_DEAD_INT_RAW RTC_CNTL_RTC_SWD_INT_RAW RTC_CNTL_RTC_SARADC2_INT_RAW RTC_CNTL_RTC_COCPU_INT_RAW RTC_CNTL_RTC_TSENS_INT_RAW RTC_CNTL_RTC_SARADC1_INT_RAW RTC_CNTL_RTC_MAIN_TIMER_INT_RAW RTC_CNTL_RTC_BROWN_OUT_INT_RAW RTC_CNTL_RTC_TOUCH_INACTIVE_INT_RAW RTC_CNTL_RTC_TOUCH_ACTIVE_INT_RAW (reserved) RTC_CNTL_RTC_ULP_CP_INT_RAW RTC_CNTL_RTC_TOUCH_SCAN_DONE_INT_RAW RTC_CNTL_SLP_REJECT_INT_RAW RTC_CNTL_SLP_WAKEUP_INT_RAW														
31																								20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0												0												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**RTC\_CNTL\_SLP\_WAKEUP\_INT\_RAW** Stores the raw interrupt triggered when the chip wakes up from sleep. (RO)

**RTC\_CNTL\_SLP\_REJECT\_INT\_RAW** Stores the raw interrupt triggered when the chip rejects to go to sleep. (RO)

**RTC\_CNTL\_RTC\_WDT\_INT\_RAW** Stores the raw RTC watchdog interrupt. (RO)

**RTC\_CNTL\_RTC\_TOUCH\_SCAN\_DONE\_INT\_RAW** Stores the raw interrupt triggered upon the completion of a touch scanning. (RO)

**RTC\_CNTL\_RTC\_ULP\_CP\_INT\_RAW** Stores the raw ULP co-processor interrupt. (RO)

**RTC\_CNTL\_RTC\_TOUCH\_DONE\_INT\_RAW** Stores the raw interrupt triggered upon the completion of a single touch. (RO)

**RTC\_CNTL\_RTC\_TOUCH\_ACTIVE\_INT\_RAW** Stores the raw interrupt triggered when a touch is detected. (RO)

**RTC\_CNTL\_RTC\_TOUCH\_INACTIVE\_INT\_RAW** Stores the raw interrupt triggered when a touch is released. (RO)

**RTC\_CNTL\_RTC\_BROWN\_OUT\_INT\_RAW** Stores the raw brownout interrupt. (RO)

**RTC\_CNTL\_RTC\_MAIN\_TIMER\_INT\_RAW** Stores the raw RTC main timer interrupt. (RO)

**RTC\_CNTL\_RTC\_SARADC1\_INT\_RAW** Stores the raw SAR ADC 1 interrupt. (RO)

**RTC\_CNTL\_RTC\_TSENS\_INT\_RAW** Stores the raw temperature sensor interrupt. (RO)

**RTC\_CNTL\_RTC\_COCPU\_INT\_RAW** Stores the raw ULP-RISCV interrupt. (RO)

**RTC\_CNTL\_RTC\_SARADC2\_INT\_RAW** Stores the raw SAR ADC 2 interrupt. (RO)

**RTC\_CNTL\_RTC\_SWD\_INT\_RAW** Stores the raw super watchdog interrupt. (RO)

**RTC\_CNTL\_RTC\_XTAL32K\_DEAD\_INT\_RAW** Stores the raw interrupt triggered when the 32 kHz crystal is dead. (RO)

Continued on the next page...

**Register 29.29: RTC\_CNTL\_INT\_RAW\_RTC\_REG (0x0044)**

Continued from the previous page...

**RTC\_CNTL\_RTC\_COCPU\_TRAP\_INT\_RAW** Stores the raw interrupt triggered when the ULP-RISCV is trapped. (RO)

**RTC\_CNTL\_RTC\_TOUCH\_TIMEOUT\_INT\_RAW** Stores the raw interrupt triggered when touch sensor times out. (RO)

**RTC\_CNTL\_RTC\_GLITCH\_DET\_INT\_RAW** Stores the raw interrupt triggered when a glitch is detected. (RO)

**Register 29.30: RTC\_CNTL\_INT\_ST\_RTC\_REG (0x0048)**

(reserved)																RTC_CNTL_RTC_GLITCH_DET_INT_ST RTC_CNTL_RTC_TOUCH_TIMEOUT_INT_ST RTC_CNTL_RTC_COCPU_TRAP_INT_ST RTC_CNTL_RTC_XTAL32K_DEAD_INT_ST RTC_CNTL_RTC_SWD_INT_ST RTC_CNTL_RTC_SARADC2_INT_ST RTC_CNTL_RTC_COCPU_INT_ST RTC_CNTL_RTC_TSENS_INT_ST RTC_CNTL_RTC_SARADC1_INT_ST RTC_CNTL_RTC_MAIN_TIMER_INT_ST RTC_CNTL_RTC_BROWN_OUT_INT_ST RTC_CNTL_RTC_TOUCH_INACTIVE_INT_ST RTC_CNTL_RTC_TOUCH_ACTIVE_INT_ST (reserved) RTC_CNTL_SLP_REJECT_INT_ST RTC_CNTL_SLP_WAKEUP_INT_ST																	
31													20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset					

**RTC\_CNTL\_SLP\_WAKEUP\_INT\_ST** Stores the status of the interrupt triggered when the chip wakes up from sleep. (RO)

**RTC\_CNTL\_SLP\_REJECT\_INT\_ST** Stores the status of the interrupt triggered when the chip rejects to go to sleep. (RO)

**RTC\_CNTL\_RTC\_WDT\_INT\_ST** Stores the status of the RTC watchdog interrupt. (RO)

**RTC\_CNTL\_RTC\_TOUCH\_SCAN\_DONE\_INT\_ST** Stores the status of the interrupt triggered upon the completion of a touch scanning. (RO)

**RTC\_CNTL\_RTC\_ULP\_CP\_INT\_ST** Stores the status of the ULP co-processor interrupt. (RO)

**RTC\_CNTL\_RTC\_TOUCH\_DONE\_INT\_ST** Stores the status of the interrupt triggered upon the completion of a single touch. (RO)

**RTC\_CNTL\_RTC\_TOUCH\_ACTIVE\_INT\_ST** Stores the status of the interrupt triggered when a touch is detected. (RO)

**RTC\_CNTL\_RTC\_TOUCH\_INACTIVE\_INT\_ST** Stores the status of the interrupt triggered when a touch is released. (RO)

**RTC\_CNTL\_RTC\_BROWN\_OUT\_INT\_ST** Stores the status of the brownout interrupt. (RO)

**RTC\_CNTL\_RTC\_MAIN\_TIMER\_INT\_ST** Stores the status of the RTC main timer interrupt. (RO)

**RTC\_CNTL\_RTC\_SARADC1\_INT\_ST** Stores the status of the SAR ADC 1 interrupt. (RO)

**RTC\_CNTL\_RTC\_TSENS\_INT\_ST** Stores the status of the temperature sensor interrupt. (RO)

**RTC\_CNTL\_RTC\_COCPU\_INT\_ST** Stores the status of the ULP-RISCV interrupt. (RO)

**RTC\_CNTL\_RTC\_SARADC2\_INT\_ST** Stores the status of the SAR ADC 2 interrupt. (RO)

**RTC\_CNTL\_RTC\_SWD\_INT\_ST** Stores the status of the super watchdog interrupt. (RO)

**RTC\_CNTL\_RTC\_XTAL32K\_DEAD\_INT\_ST** Stores the status of the interrupt triggered when the 32 kHz crystal is dead. (RO)

Continued on the next page...

**Register 29.30: RTC\_CNTL\_INT\_ST\_RTC\_REG (0x0048)**

Continued from the previous page...

**RTC\_CNTL\_RTC\_COCPU\_TRAP\_INT\_ST** Stores the status of the interrupt triggered when the ULP-RISCV is trapped. (RO)

**RTC\_CNTL\_RTC\_TOUCH\_TIMEOUT\_INT\_ST** Stores the status of the interrupt triggered when touch sensor times out. (RO)

**RTC\_CNTL\_RTC\_GLITCH\_DET\_INT\_ST** Stores the status of the interrupt triggered when a glitch is detected. (RO)

Register 29.31: RTC\_CNTL\_INT\_CLR\_RTC\_REG (0x004C)

(reserved)												RTC_CNTL_RTC_GLITCH_DET_INT_CLR RTC_CNTL_RTC_TOUCH_TIMEOUT_INT_CLR RTC_CNTL_RTC_COCPU_TRAP_INT_CLR RTC_CNTL_RTC_XTAL32K_DEAD_INT_CLR RTC_CNTL_RTC_SWD_INT_CLR RTC_CNTL_RTC_SARADC2_INT_CLR RTC_CNTL_RTC_COCPU_INT_CLR RTC_CNTL_RTC_TSENS_INT_CLR RTC_CNTL_RTC_SARADC1_INT_CLR RTC_CNTL_RTC_MAIN_TIMER_INT_CLR RTC_CNTL_RTC_BROWN_OUT_INT_CLR RTC_CNTL_RTC_TOUCH_INACTIVE_INT_CLR RTC_CNTL_RTC_TOUCH_ACTIVE_INT_CLR RTC_CNTL_RTC_ULP_CP_INT_CLR (reserved) RTC_CNTL_SLP_REJECT_INT_CLR RTC_CNTL_SLP_WAKEUP_INT_CLR																					
31													20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset				

**RTC\_CNTL\_SLP\_WAKEUP\_INT\_CLR** Clears the interrupt triggered when the chip wakes up from sleep. (WO)

**RTC\_CNTL\_SLP\_REJECT\_INT\_CLR** Clears the interrupt triggered when the chip rejects to go to sleep. (WO)

**RTC\_CNTL\_RTC\_WDT\_INT\_CLR** Clears the RTC watchdog interrupt. (WO)

**RTC\_CNTL\_RTC\_TOUCH\_SCAN\_DONE\_INT\_CLR** Clears the interrupt triggered upon the completion of a touch scanning. (WO)

**RTC\_CNTL\_RTC\_ULP\_CP\_INT\_CLR** Clears the ULP co-processor interrupt. (WO)

**RTC\_CNTL\_RTC\_TOUCH\_DONE\_INT\_CLR** Clears the interrupt triggered upon the completion of a single touch. (WO)

**RTC\_CNTL\_RTC\_TOUCH\_ACTIVE\_INT\_CLR** Clears the interrupt triggered when a touch is detected. (WO)

**RTC\_CNTL\_RTC\_TOUCH\_INACTIVE\_INT\_CLR** Clears the interrupt triggered when a touch is released. (WO)

**RTC\_CNTL\_RTC\_BROWN\_OUT\_INT\_CLR** Clears the brownout interrupt. (WO)

**RTC\_CNTL\_RTC\_MAIN\_TIMER\_INT\_CLR** Clears the RTC main timer interrupt. (WO)

**RTC\_CNTL\_RTC\_SARADC1\_INT\_CLR** Clears the SAR ADC 1 interrupt. (WO)

**RTC\_CNTL\_RTC\_TSENS\_INT\_CLR** Clears the temperature sensor interrupt. (WO)

**RTC\_CNTL\_RTC\_COCPU\_INT\_CLR** Clears the ULP-RISCV interrupt. (WO)

**RTC\_CNTL\_RTC\_SARADC2\_INT\_CLR** Clears the SAR ADC 2 interrupt. (WO)

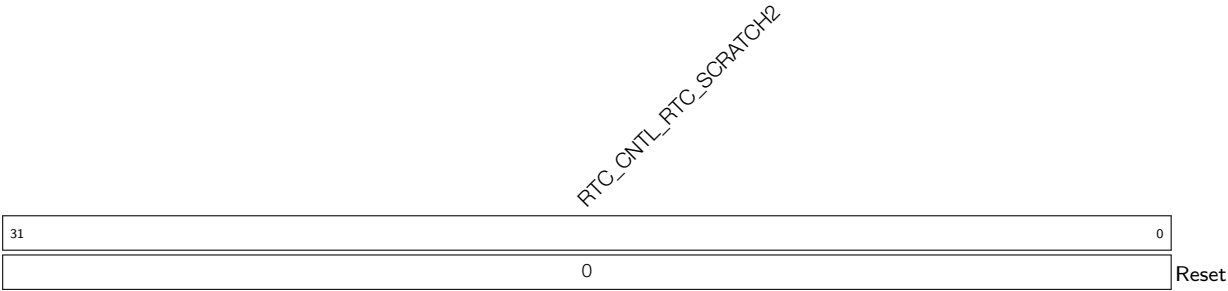
**RTC\_CNTL\_RTC\_SWD\_INT\_CLR** Clears the super watchdog interrupt. (WO)

**RTC\_CNTL\_RTC\_XTAL32K\_DEAD\_INT\_CLR** Clears the interrupt triggered when the 32 kHz crystal is dead. (WO)

Continued on the next page...

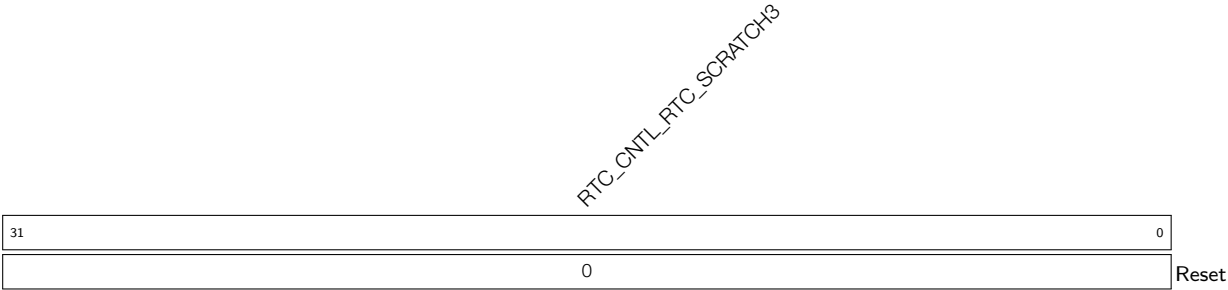


Register 29.34: RTC\_CNTL\_STORE2\_REG (0x0058)



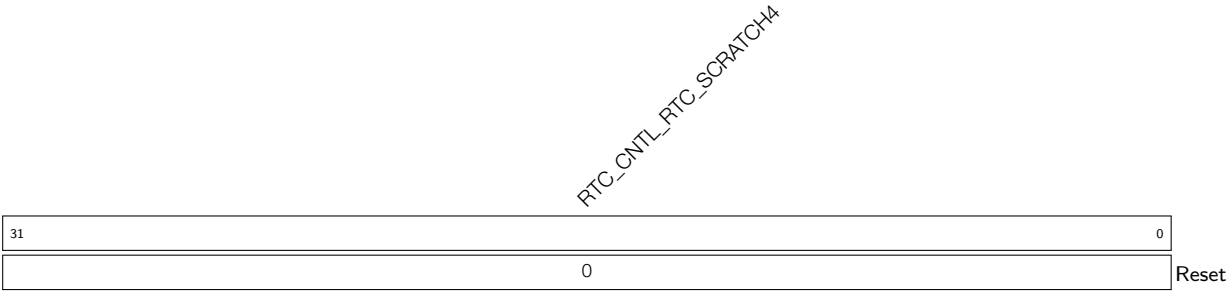
RTC\_CNTL\_RTC\_SCRATCH2 Reservation register 2. (R/W)

Register 29.35: RTC\_CNTL\_STORE3\_REG (0x005C)



RTC\_CNTL\_RTC\_SCRATCH3 Reservation register 3. (R/W)

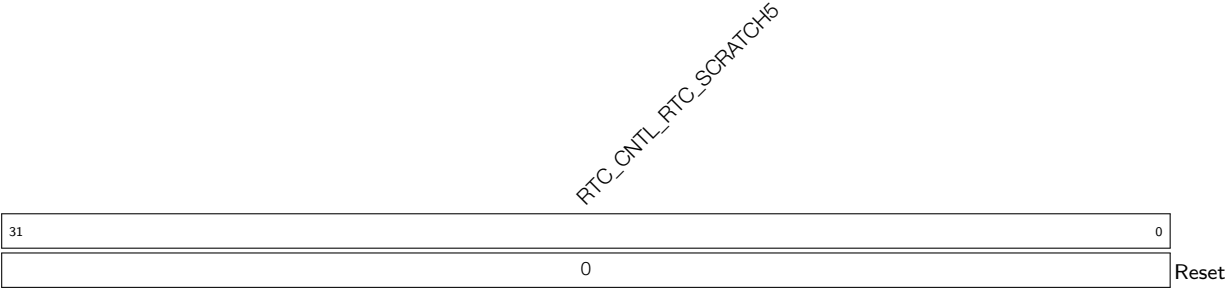
Register 29.36: RTC\_CNTL\_STORE4\_REG (0x00BC)



RTC\_CNTL\_RTC\_SCRATCH4 Reservation register 4. (R/W)

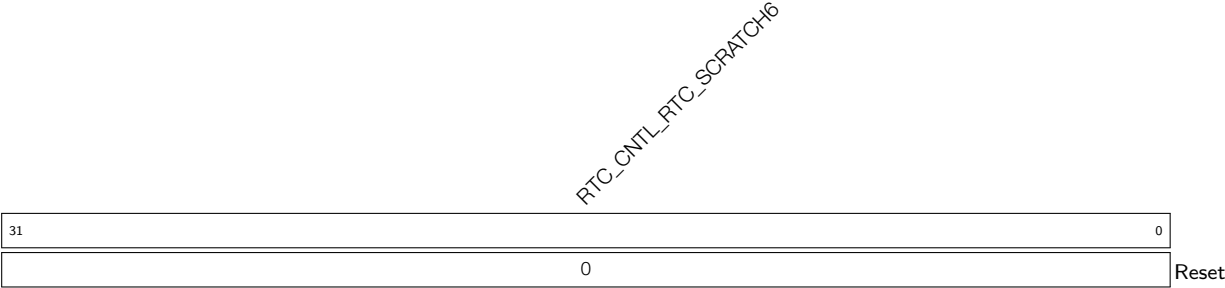


Register 29.37: RTC\_CNTL\_STORE5\_REG (0x00C0)



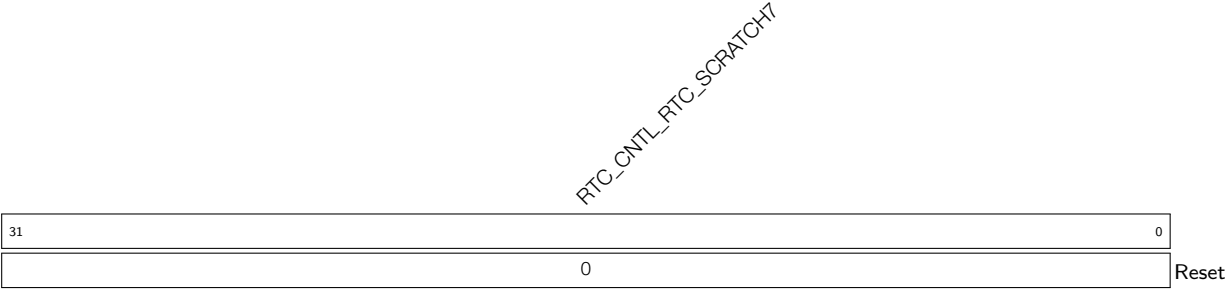
**RTC\_CNTL\_RTC\_SCRATCH5** Reservation register 5. (R/W)

Register 29.38: RTC\_CNTL\_STORE6\_REG (0x00C4)



**RTC\_CNTL\_RTC\_SCRATCH6** Reservation register 6. (R/W)

Register 29.39: RTC\_CNTL\_STORE7\_REG (0x00C8)



**RTC\_CNTL\_RTC\_SCRATCH7** Reservation register 7. (R/W)

Register 29.40: RTC\_CNTL\_EXT\_XTL\_CONF\_REG (0x0060)

RTC_CNTL_XTL_EXT_CTR_EN																RTC_CNTL_RTC_XTAL32K_GPIO_SEL																RTC_CNTL_ENCKINIT_XTAL_32K																RTC_CNTL_XTAL32K_XPD_FORCE																RTC_CNTL_XTAL32K_AUTO_RETURN																RTC_CNTL_XTAL32K_AUTO_RESTART																RTC_CNTL_XTAL32K_EXT_CLK_FO																RTC_CNTL_XTAL32K_WDT_RESET																RTC_CNTL_XTAL32K_WDT_EN																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
(reserved)																(reserved)																(reserved)																(reserved)																(reserved)																(reserved)																(reserved)																(reserved)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
31	30																									24	23	22															19															9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
0	0	0	0	0	0	0	0	0	0	0	0x0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**RTC\_CNTL\_XTAL32K\_WDT\_EN** Set this bit to enable the 32 kHz crystal watchdog. (R/W)

**RTC\_CNTL\_XTAL32K\_WDT\_CLK\_FO** Set this bit to FPU the 32 kHz crystal watchdog clock. (R/W)

**RTC\_CNTL\_XTAL32K\_WDT\_RESET** Set this bit to reset the 32 kHz crystal watchdog by SW. (R/W)

**RTC\_CNTL\_XTAL32K\_EXT\_CLK\_FO** Set this bit to FPU the external clock of 32 kHz crystal. (R/W)

**RTC\_CNTL\_XTAL32K\_AUTO\_BACKUP** Set this bit to switch to the backup clock when the 32 kHz crystal is dead. (R/W)

**RTC\_CNTL\_XTAL32K\_AUTO\_RESTART** Set this bit to restart the 32 kHz crystal automatically when the 32 kHz crystal is dead. (R/W)

**RTC\_CNTL\_XTAL32K\_AUTO\_RETURN** Set this bit to switch back to 32 kHz crystal when the 32 kHz crystal is restarted. (R/W)

**RTC\_CNTL\_XTAL32K\_XPD\_FORCE** Set this bit to allow the software to FPD the 32 kHz crystal; Reset this bit to allow the FSM to FPD the 32 kHz crystal. (R/W)

**RTC\_CNTL\_ENCKINIT\_XTAL\_32K** Set 1 to apply an internal clock to help the 32 kHz crystal to start. (R/W)

**RTC\_CNTL\_RTC\_WDT\_STATE** Stores the status of the 32 kHz watchdog. (RO)

**RTC\_CNTL\_RTC\_XTAL32K\_GPIO\_SEL** Selects the 32 kHz crystal clock. 0: selects the external 32 kHz clock; 1: selects clock from the RTC GPIO X32P\_C. (R/W)

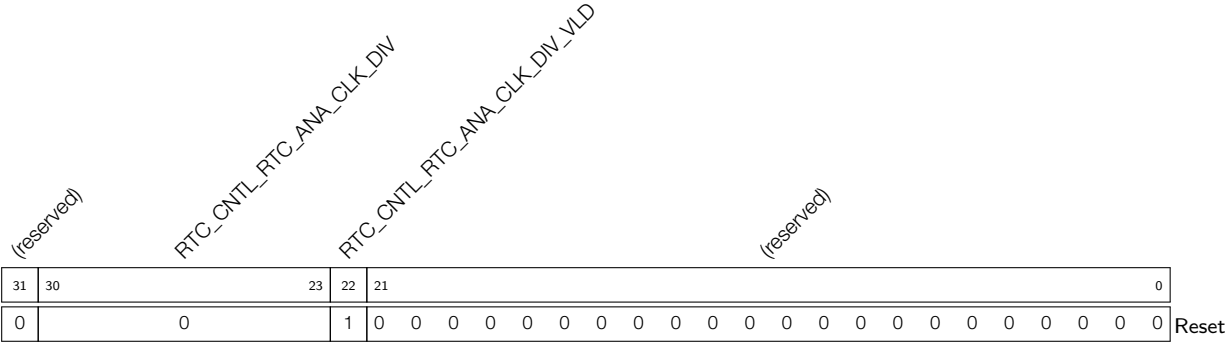
**RTC\_CNTL\_XTL\_EXT\_CTR\_LV** 0: powers down XTAL at high level; 1: powers down XTAL at low level. (R/W)

**RTC\_CNTL\_XTL\_EXT\_CTR\_EN** Enables the GPIO to power down the crystal oscillator. (R/W)

## 691

ESP32-S2 TRM (Preliminary V0.5)

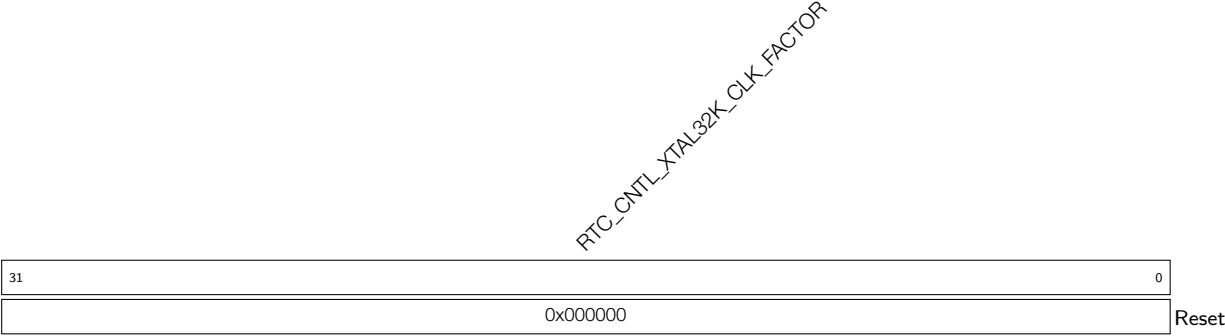
Register 29.42: RTC\_CNTL\_SLOW\_CLK\_CONF\_REG (0x0078)



**RTC\_CNTL\_RTC\_ANA\_CLK\_DIV\_VLD** Synchronizes the reg\_rtc\_ana\_clk\_div. Note that you have to invalidate the bus before modifying the frequency divider, and then validate the new divider clock. (R/W)

**RTC\_CNTL\_RTC\_ANA\_CLK\_DIV** Set the divider for the RTC clock. (R/W)

Register 29.43: RTC\_CNTL\_XTAL32K\_CLK\_FACTOR\_REG (0x00F0)



**RTC\_CNTL\_XTAL32K\_CLK\_FACTOR** Configures the divider factor for the 32 kHz crystal oscillator. (R/W)

Register 29.44: RTC\_CNTL\_XTAL32K\_CONF\_REG (0x00F4)

RTC_CNTL_XTAL32K_STABLE_THRES				RTC_CNTL_XTAL32K_WDT_TIMEOUT				RTC_CNTL_XTAL32K_RESTART_WAIT				RTC_CNTL_XTAL32K_RETURN_WAIT				
31	28	27	20				19	4				3	0			
0x0				0xff				0x00				0x0				Reset

**RTC\_CNTL\_XTAL32K\_RETURN\_WAIT** Defines the waiting cycles before returning to the normal 32 kHz crystal oscillator. (R/W)

**RTC\_CNTL\_XTAL32K\_RESTART\_WAIT** Defines the waiting cycles before restarting the 32 kHz crystal oscillator. (R/W)

**RTC\_CNTL\_XTAL32K\_WDT\_TIMEOUT** Defines the waiting period for clock detection. If no clock is detected after this period, the 32 kHz crystal oscillator can be regarded as dead. (R/W)

**RTC\_CNTL\_XTAL32K\_STABLE\_THRES** Defines the allowed restarting period, within which the 32 kHz crystal oscillator can be regarded as stable. (R/W)

Register 29.45: RTC\_CNTL\_WDTCONFIG0\_REG (0x0094)

RTC_CNTL_WDT_EN		RTC_CNTL_WDT_STG0		RTC_CNTL_WDT_STG1		RTC_CNTL_WDT_STG2		RTC_CNTL_WDT_STG3		RTC_CNTL_WDT_CPU_RESET_LENGTH		RTC_CNTL_WDT_SYS_RESET_LENGTH		RTC_CNTL_WDT_FLASHBOOT_MOD_EN		RTC_CNTL_WDT_PROCPU_RESET_EN		(reserved)		(reserved)												
31	30	28	27	25	24	22	21	19	18	16	15	13	12	11	10	9	8															0
0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x1	0x1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset		

**RTC\_CNTL\_WDT\_PAUSE\_IN\_SLP** Set this bit to pause the watchdog in sleep. (R/W)

**RTC\_CNTL\_WDT\_PROCPU\_RESET\_EN** Set this bit to allow the watchdog to be able to reset CPU. (R/W)

**RTC\_CNTL\_WDT\_FLASHBOOT\_MOD\_EN** Set this bit to enable watchdog when the chip boots from flash. (R/W)

**RTC\_CNTL\_WDT\_SYS\_RESET\_LENGTH** Sets the length of the system reset counter. (R/W)

**RTC\_CNTL\_WDT\_CPU\_RESET\_LENGTH** Sets the length of the CPU reset counter. (R/W)

**RTC\_CNTL\_WDT\_STG3** 1: enable at the interrupt stage, 2: enable at the CPU stage, 3: enable at the system stage, 4: enable at the system and RTC stage. (R/W)

**RTC\_CNTL\_WDT\_STG2** 1: enable at the interrupt stage, 2: enable at the CPU stage, 3: enable at the system stage, 4: enable at the system and RTC stage. (R/W)

**RTC\_CNTL\_WDT\_STG1** 1: enable at the interrupt stage, 2: enable at the CPU stage, 3: enable at the system stage, 4: enable at the system and RTC stage. (R/W)

**RTC\_CNTL\_WDT\_STG0** 1: enable at the interrupt stage, 2: enable at the CPU stage, 3: enable at the system stage, 4: enable at the system and RTC stage. (R/W)

**RTC\_CNTL\_WDT\_EN** Set this bit to enable the RTC watchdog. (R/W)

**Note:**

For details, please refer to Chapter [14 XTAL32K Watchdog](#).

**Register 29.46: RTC\_CNTL\_WDTCONFIG1\_REG (0x0098)**

<i>RTC_CNTL_WDT_STG0_HOLD</i>	
31	0
200000	
	Reset

**RTC\_CNTL\_WDT\_STG0\_HOLD** Configures the hold time of RTC watchdog at level 1. (R/W)

**Register 29.47: RTC\_CNTL\_WDTCONFIG2\_REG (0x009C)**

<i>RTC_CNTL_WDT_STG1_HOLD</i>	
31	0
80000	
	Reset

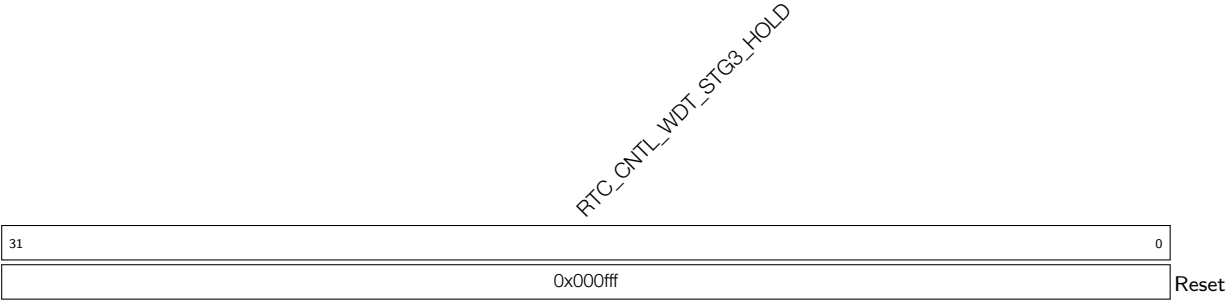
**RTC\_CNTL\_WDT\_STG1\_HOLD** Configures the hold time of RTC watchdog at level 2. (R/W)

**Register 29.48: RTC\_CNTL\_WDTCONFIG3\_REG (0x00A0)**

<i>RTC_CNTL_WDT_STG2_HOLD</i>	
31	0
0x000fff	
	Reset

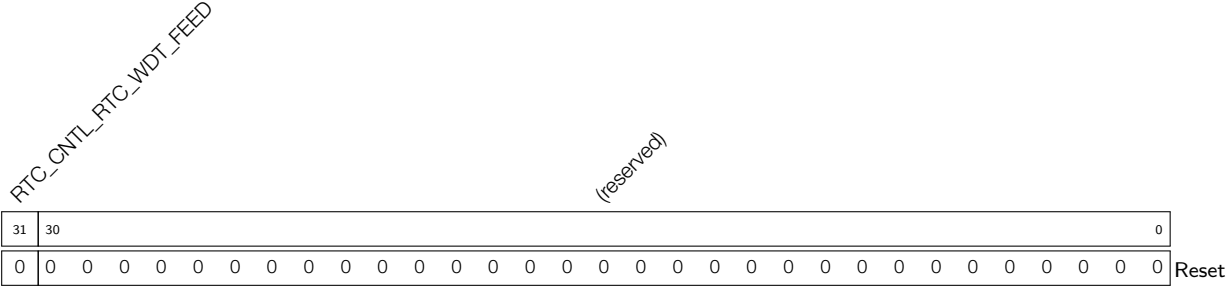
**RTC\_CNTL\_WDT\_STG2\_HOLD** Configures the hold time of RTC watchdog at level 3. (R/W)

Register 29.49: RTC\_CNTL\_WDTCONFIG4\_REG (0x00A4)



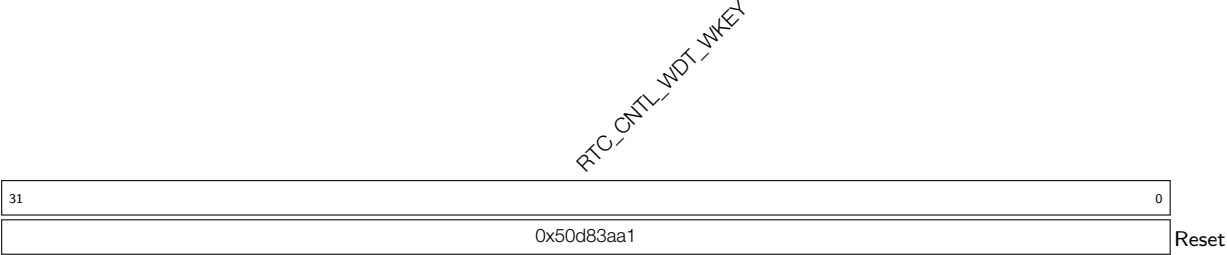
**RTC\_CNTL\_WDT\_STG3\_HOLD** Configures the hold time of RTC watchdog at level 4. (R/W)

Register 29.50: RTC\_CNTL\_WDTFEED\_REG (0x00A8)



**RTC\_CNTL\_RTC\_WDT\_FEED** Set this bit to feed the RTC watchdog. (WO)

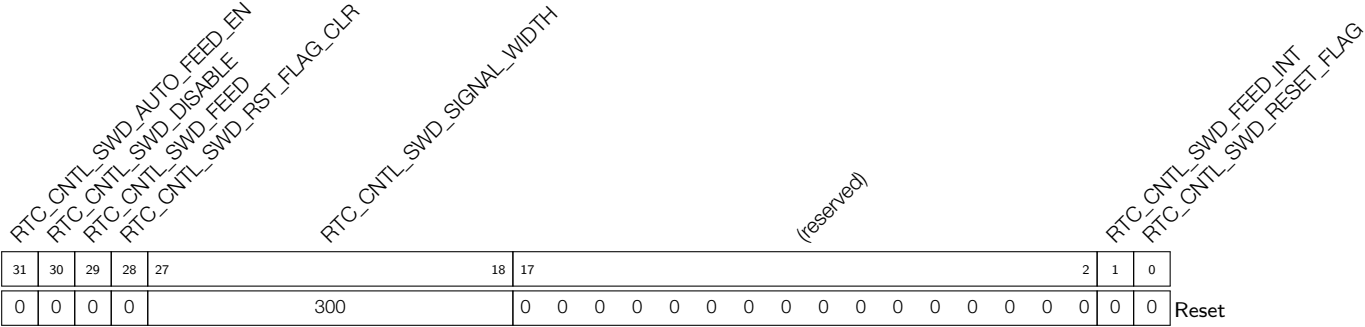
Register 29.51: RTC\_CNTL\_WDTWPROTECT\_REG (0x00AC)



**RTC\_CNTL\_WDT\_WKEY** Sets the write protection key of the watchdog. (R/W)

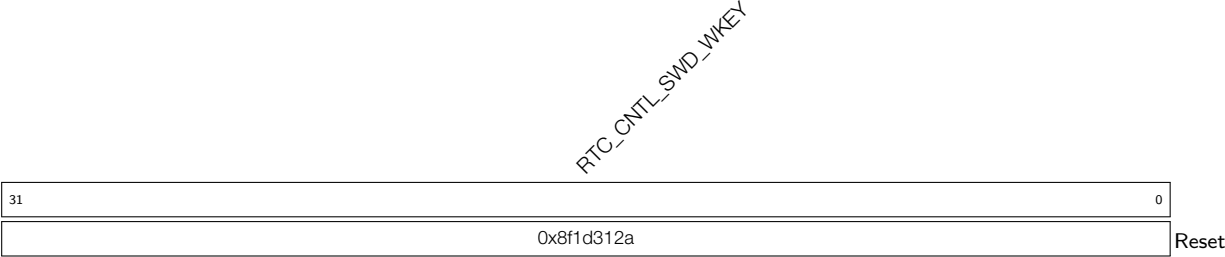


Register 29.52: RTC\_CNTL\_SWD\_CONF\_REG (0x00B0)



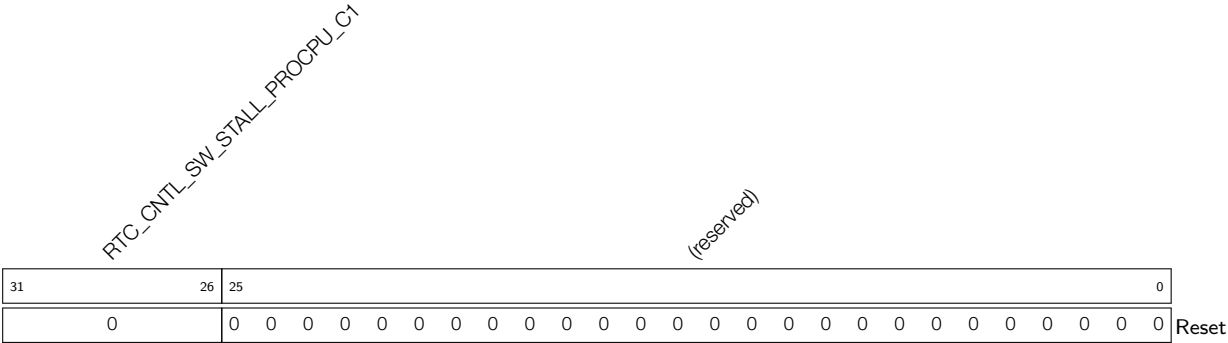
- RTC\_CNTL\_SWD\_RESET\_FLAG** Indicates the super watchdog reset flag. (RO)
- RTC\_CNTL\_SWD\_FEED\_INT** Receiving this interrupt leads to feeding the super watchdog via SW. (RO)
- RTC\_CNTL\_SWD\_SIGNAL\_WIDTH** Adjusts the signal width sent to the super watchdog. (R/W)
- RTC\_CNTL\_SWD\_RST\_FLAG\_CLR** Set to reset the super watchdog reset flag. (WO)
- RTC\_CNTL\_SWD\_FEED** Set to feed the super watchdog via SW. (WO)
- RTC\_CNTL\_SWD\_DISABLE** Set this bit to disable super watchdog. (R/W)
- RTC\_CNTL\_SWD\_AUTO\_FEED\_EN** Set this bit to enable automatic watchdog feeding upon interrupts. (R/W)

Register 29.53: RTC\_CNTL\_SWD\_WPROTECT\_REG (0x00B4)



- RTC\_CNTL\_SWD\_WKEY** Sets the write protection key of the super watchdog. (R/W)

Register 29.54: RTC\_CNTL\_SW\_CPU\_STALL\_REG (0x00B8)



**RTC\_CNTL\_SW\_STALL\_PROCPU\_C1** When [RTC\\_CNTL\\_SW\\_STALL\\_PROCPU\\_C0](#) is configured to 0x2, setting this bit to 0x21 stalls the CPU by SW. (R/W)

Register 29.55: RTC\_CNTL\_PAD\_HOLD\_REG (0x00D4)

(reserved)										<div>RTC_CNTL_RTC_PAD21_HOLD</div> <div>RTC_CNTL_RTC_PAD20_HOLD</div> <div>RTC_CNTL_RTC_PAD19_HOLD</div> <div>RTC_CNTL_PDAC2_HOLD</div> <div>RTC_CNTL_PDAC1_HOLD</div> <div>RTC_CNTL_X32N_HOLD</div> <div>RTC_CNTL_X32P_HOLD</div> <div>RTC_CNTL_TOUCH_PAD14_HOLD</div> <div>RTC_CNTL_TOUCH_PAD13_HOLD</div> <div>RTC_CNTL_TOUCH_PAD12_HOLD</div> <div>RTC_CNTL_TOUCH_PAD11_HOLD</div> <div>RTC_CNTL_TOUCH_PAD10_HOLD</div> <div>RTC_CNTL_TOUCH_PAD9_HOLD</div> <div>RTC_CNTL_TOUCH_PAD8_HOLD</div> <div>RTC_CNTL_TOUCH_PAD7_HOLD</div> <div>RTC_CNTL_TOUCH_PAD6_HOLD</div> <div>RTC_CNTL_TOUCH_PAD5_HOLD</div> <div>RTC_CNTL_TOUCH_PAD4_HOLD</div> <div>RTC_CNTL_TOUCH_PAD3_HOLD</div> <div>RTC_CNTL_TOUCH_PAD2_HOLD</div> <div>RTC_CNTL_TOUCH_PAD0_HOLD</div>																						
31										22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

**RTC\_CNTL\_TOUCH\_PAD0\_HOLD** Sets the touch GPIO 0 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD1\_HOLD** Sets the touch GPIO 1 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD2\_HOLD** Sets the touch GPIO 2 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD3\_HOLD** Sets the touch GPIO 3 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD4\_HOLD** Sets the touch GPIO 4 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD5\_HOLD** Sets the touch GPIO 5 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD6\_HOLD** Sets the touch GPIO 6 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD7\_HOLD** Sets the touch GPIO 7 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD8\_HOLD** Sets the touch GPIO 8 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD9\_HOLD** Sets the touch GPIO 9 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD10\_HOLD** Sets the touch GPIO 10 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD11\_HOLD** Sets the touch GPIO 11 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD12\_HOLD** Sets the touch GPIO 12 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD13\_HOLD** Sets the touch GPIO 13 to the holding state. (R/W)

**RTC\_CNTL\_TOUCH\_PAD14\_HOLD** Sets the touch GPIO 14 to the holding state. (R/W)

**RTC\_CNTL\_X32P\_HOLD** Sets the x32p to the holding state. (R/W)

**RTC\_CNTL\_X32N\_HOLD** Sets the x32n to the holding state. (R/W)

**RTC\_CNTL\_PDAC1\_HOLD** Sets the pdac1 to the holding state. (R/W)

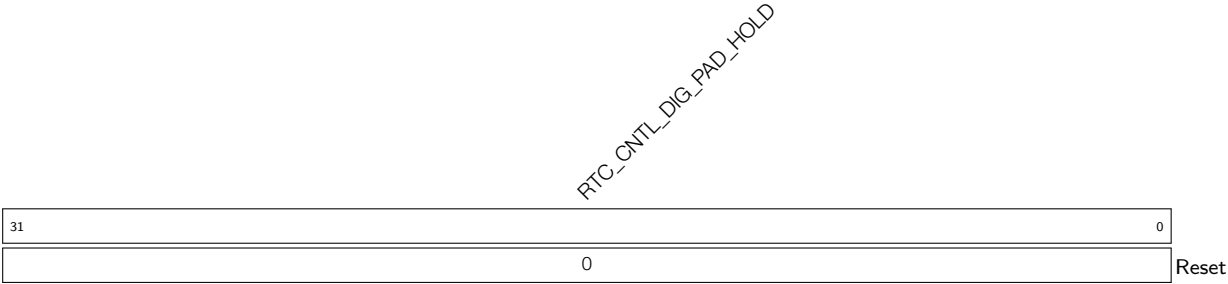
**RTC\_CNTL\_PDAC2\_HOLD** Sets the pdac2 to the holding state. (R/W)

**RTC\_CNTL\_RTC\_PAD19\_HOLD** Sets the RTG GPIO 19 to the holding state. (R/W)

**RTC\_CNTL\_RTC\_PAD20\_HOLD** Sets the RTG GPIO 20 to the holding state. (R/W)

**RTC\_CNTL\_RTC\_PAD21\_HOLD** Sets the RTG GPIO 21 to the holding state. (R/W)

Register 29.56: RTC\_CNTL\_DIG\_PAD\_HOLD\_REG (0x00D8)



**RTC\_CNTL\_DIG\_PAD\_HOLD** Set GPIO 21 to GPIO 45 to the holding state. (See bitmap to locate any GPIO). (R/W)

Register 29.57: RTC\_CNTL\_BROWN\_OUT\_REG (0x00E4)

RTC_CNTL_RTC_BROWN_OUT_DET								RTC_CNTL_BROWN_OUT_RST_WAIT								RTC_CNTL_BROWN_OUT_PD_RF_ENA								RTC_CNTL_BROWN_OUT_CLOSE_FLASH_ENA													
RTC_CNTL_BROWN_OUT_ENA								RTC_CNTL_BROWN_OUT_RST_ENA								(reserved)								RTC_CNTL_BROWN_OUT_INT_WAIT													
RTC_CNTL_BROWN_OUT_CNT_CLR								RTC_CNTL_BROWN_OUT_RST_SEL								(reserved)								RTC_CNTL_BROWN_OUT2_ENA													
31	30	29	28	27	26	25											16	15	14	13											4	3				1	0
0	0	0	0	0	0	0	0x3ff										0	0	0x2ff										0	0	0	1	Reset				

**RTC\_CNTL\_BROWN\_OUT2\_ENA** Enables the brown\_out2 to initiate a chip reset. (R/W)

**RTC\_CNTL\_BROWN\_OUT\_INT\_WAIT** Configures the waiting cycles before sending an interrupt. (R/W)

**RTC\_CNTL\_BROWN\_OUT\_CLOSE\_FLASH\_ENA** Set this bit to enable PD the flash when a brown-out happens. (R/W)

**RTC\_CNTL\_BROWN\_OUT\_PD\_RF\_ENA** Set this bit to enable PD the RF circuits when a brown-out happens. (R/W)

**RTC\_CNTL\_BROWN\_OUT\_RST\_WAIT** Configures the waiting cycles before the reset after a brown-out. (R/W)

**RTC\_CNTL\_BROWN\_OUT\_RST\_ENA** Enables to reset brown-out. (R/W)

**RTC\_CNTL\_BROWN\_OUT\_RST\_SEL** Selects the reset type when a brown-out happens. 1: chip reset, 0: system reset. (R/W)

**RTC\_CNTL\_BROWN\_OUT\_CNT\_CLR** Clears the brown-out counter. (WO)

**RTC\_CNTL\_BROWN\_OUT\_ENA** Set this bit to enable brown-out detection. (R/W)

**RTC\_CNTL\_RTC\_BROWN\_OUT\_DET** Indicates the status of the brown-out signal. (RO)

## Revision History

Date	Version	Release notes
2020-06-19	V0.5	<p>Added Chapter <a href="#">27 On-Chip Sensor and Analog Signal Processing</a></p> <p>Updated the following chapters:</p> <ul style="list-style-type: none"> <li>Added information about Copy DMA and Crypto DMA, and updated information about Internal DMA and EMDA in Chapter <a href="#">7 DMA Controller</a></li> <li>Added Section <a href="#">13.4 Super Watchdog</a> in Chapter <a href="#">13 Watchdog Timers</a></li> </ul>
2020-05-21	V0.4	<p>Added the following chapters:</p> <ul style="list-style-type: none"> <li>Chapter <a href="#">18 TWAI</a></li> <li>Chapter <a href="#">29 Low-power Management</a></li> </ul> <p>Updated the following chapters:</p> <ul style="list-style-type: none"> <li>Added a new section <a href="#">5.4 Dedicated GPIO</a> in Chapter <a href="#">5 IO MUX and GPIO Matrix</a></li> <li>Removed bit EFUSE_PGM_DATA1_REG[16] from the list of bits that are reserved and can only be used by software; added bit <a href="#">EFUSE_RPT4_RESERVED5</a> and <a href="#">EFUSE_RPT4_RESERVED5_ERR</a> in Chapter <a href="#">16 eFuse Controller</a></li> <li>Updated the base address in Chapter <a href="#">22 Random Number Generator</a></li> </ul>
2020-04-01	V0.3	<p>Added the following chapters:</p> <ul style="list-style-type: none"> <li>Chapter <a href="#">14 XTAL32K Watchdog</a></li> <li>Chapter <a href="#">15 System Timer</a></li> <li>Chapter <a href="#">24 Permission Control</a></li> <li>Chapter <a href="#">26 HMAC Module</a></li> <li>Chapter <a href="#">28 ULP Coprocessor</a></li> </ul> <p>Updated RTC_CLK frequency in Chapter <a href="#">2 Reset and Clock</a></p>
2020-01-20	V0.2	<p>Added the following chapters:</p> <ul style="list-style-type: none"> <li>Chapter <a href="#">5 IO MUX and GPIO Matrix</a></li> <li>Chapter <a href="#">7 DMA Controller</a></li> <li>Chapter <a href="#">8 UART Controller</a></li> </ul> <p>Updated the configurations of eFuse-programming and eFuse-read timing parameters in Chapter <a href="#">16 eFuse Controller</a></p>
2019-11-27	V0.1	Preliminary release