

WM_W800_Register Manual

V2.1

Beijing Lianshengde Microelectronics Co., Ltd. (winner micro)

Address: 18th Floor, Yindu Building, No. 67, Fucheng Road, Haidian District, Beijing

Tel: +86-10-62161900

Company website: www.winnermicro.com

Document modification record

Version	revision time	revision history	author	audit
V0.1	2019-09-25	create	chengsheng	
V0.2	2020-05-13	Revised the legacy description error;	Chengsheng	
V1.0	2020-7-20	Update digital function interface	Ray	
V2.0	2020-8-20	Improve high-speed interface function	Ray	
V2.1	2020-09-08	Added description of SD_ADC module	chengsheng	

Table of contents

Document Modification History	2
Table of contents.....	3
Figure Catalog	twenty two
Table Catalog	twenty three
1 Introduction	34
1.1 Purpose of writing.....	34
1.2 References	34
2 Features	35
3 Overview.....	39
4 Chip structure.....	41
4.1 Chip structure.....	41
4.2 Bus structure.....	43
4.3 Clock Structure47
4.4 Address space.....	48
4.4.1 SRAM	53
4.4.2 Flash.....	53
4.4.3 PSRAM.....	54
4.5 Startup configuration.....	.54
5 Clock and Reset Module	56
5.1 Function overview.....	.56
5.2 Main features.....	.56

5.3 Function description.....	56
5.3.1 Clock Gating	56
5.3.2 Clock Adaptive Shutdown.....	57
5.3.3 Function reset.....	57
5.3.4 Clock Divide	57
5.3.5 Debug function control.....	59
5.4 Register Description.....	60
5.4.1 Register List	60
5.4.2 Software Clock Gating Enable Register.....	60
5.4.3 Software Clock Mask Register	64
5.4.4 Software reset control register.....	65
5.4.5 Clock Divider Configuration Register.....	71
5.4.6 Debug Control Registers.....	73
5.4.7 I2S clock control register.....	74
5.4.8 Reset Status Register	76
6 DMA module.....	77
6.1 Function overview.....	77
6.2 Main Features	77
6.3 Function description.....	77
6.3.1 DMA channel.....	77
6.3.2 DMA data flow.....	78
6.3.3 DMA Cycling Mode.....	79

6.3.4 DMA transfer mode.....	79
6.3.5 DMA Peripheral Selection	79
6.3.6 DMA linked list mode.....	80
6.3.7 DMA Interrupt.....	80
6.4 Register description.....	80
6.4.1 Register List	80
6.4.2 Interrupt Mask Register	82
6.4.3 Interrupt Status Register	83
6.4.4 UART selection register.....	84
6.4.5 DMA Source Address Register.....	85
6.4.6 DMA Destination Address Register.....	85
6.4.7 DMA loop source start address register.....	85
6.4.8 DMA loop destination start address register.....	86
6.4.9 DMA Cycle Length Register	86
6.4.10 DMA Channel Control Register.....	86
6.4.11 DMA mode selection register.....	87
6.4.12 DMA data flow control register.....	88
6.4.13 DMA transfer bytes register.....	90
6.4.14 DMA linked list entry address register.....	90
6.4.15 DMA current destination address register.....	90
7 Generic Hardware Encryption Module	92
7.1 Function overview.....	92

7.2 Main features.....	92
7.3 Function description.....	92
7.3.1 SHA1 encryption.....	92
7.3.2 MD5 encryption.....	92
7.3.3 RC4 encryption.....	93
7.3.4 DES encryption.....	93
7.3.5 3DES encryption.....	93
7.3.6 AES encryption.....	93
7.3.7 CRC encryption.....	93
7.3.8 TRNG random number generator.....	94
7.4 Register Description.....	95
7.4.1 Register List	95
7.4.2 Configuration Registers.....	96
7.4.3 TRNG Control Register	99
7.4.4 Control Registers.....	100
7.4.5 Status Register	101
8 RSA encryption module.....	102
8.1 Function overview.....	102
8.2 Main features.....	102
8.3 Function description.....	102
8.3.1 Modular multiplication function	102
8.4 Register description.....	102

8.4.1 Register List	102
8.4.2 Data X Register	103
8.4.3 Data Y register.....	103
8.4.4 Data M register.....	103
8.4.5 Data D Register	103
8.4.6 RSA Control Register	104
8.4.7 Parameter MC register.....	105
8.4.8 Parameter N register.....	105
9 GPIO module.....	106
9.1 Function overview.....	106
9.2 Main Features	106
9.3 Function description.....	106
9.4 Register Description.....	107
9.4.1 Register List	107
9.4.2 GPIO data register.....	109
9.4.3 GPIO data enable register.....	110
9.4.4 GPIO direction control register.....	110
9.4.5 GPIO pull-up and pull-down control register.....	111
9.4.6 GPIO multiplexing selection register.....	112
9.4.7 GPIO multiplexing selection register 1.....	114
9.4.8 GPIO multiplexing selection register 0.....	114
9.4.9 GPIO interrupt trigger mode configuration register.....	115

9.4.10 GPIO interrupt edge-triggered mode configuration register.....	116
9.4.11 GPIO interrupt upper and lower edge trigger configuration register.....	116
9.4.12 GPIO Interrupt Enable Configuration Register.....	117
9.4.13 GPIO Raw Interrupt Status Register.....	118
9.4.14 GPIO Masked Interrupt Status Register.....	118
9.4.15 GPIO Interrupt Clear Control Register.....	119
10 High-speed SPI device controller.....	120
10.1 Function overview.....	120
10.2 Main Features	120
10.3 Function description.....	120
10.3.1 Introduction to the SPI protocol.....	120
10.3.2 SPI working process.....	121
10.4 Register Description.....	121
10.4.1 List of registers for internal operation of HSPI chip.....	121
10.4.2 Host access to HSPI controller register list.....	125
10.4.3 High Speed SPI Device Controller Interface Timing.....	130
11 SDIO device controller.....	141
11.1 Function overview.....	141
11.2 Main Features	141
11.3 Function description.....	141
11.3.1 SDIO bus.....	141
11.3.2 SDIO Commands.....	142

11.3.3 SDIO internal storage.....	142
11.4 Register Description.....	144
11.4.1 Register List	144
11.4.2 SDIO Fn0 register.....	144
11.4.3 SDIO Fn1 register.....	157
12 HSPI/SDIO Wrapper Controller.....	168
12.1 Function overview.....	168
12.2 Main Features	168
12.3 Function description.....	169
12.3.1 Uplink data receiving function.....	169
12.3.2 Downlink data transfer function.....	170
12.4 Register Description.....	170
12.4.1 Register List	170
12.4.2 WRAPPER INTERRUPT STATUS REGISTER	172
12.4.3 WRAPPER INTERRUPT CONFIGURATION REGISTER	172
12.4.4 WRAPPER Upstream Command Ready Register.....	172
12.4.5 WRAPPER downlink command buf ready register.....	173
12.4.6 SDIO TX Link Enable Register.....	173
12.4.7 SDIO TX Link Address Register.....	174
12.4.8 SDIO TX enable register.....	174
12.4.9 SDIO TX Status Register	174
12.4.10 SDIO RX Link Enable Register.....	175

12.4.11 SDIO RX Link Address Register.....	175
12.4.12 SDIO RX Enable Register.....	176
12.4.13 SDIO RX Status Register	176
12.4.14 WRAPPER CMD BUF Base Address Register.....	177
12.4.15 WRAPPER CMD BUF SIZE register.....	177
13 SDIO HOST device controller.....	178
13.1 Function overview.....	178
13.2 Main Features	178
13.3 Function description.....	178
13.4 Register Description.....	179
13.4.1 Register List	179
14 SPI Controller.....	200
14.1 Function overview.....	200
14.2 Main Features	200
14.3 Function description.....	200
14.3.1 Master-slave configuration	200
14.3.2 Multiple Mode Support.....	201
14.3.3 Efficient data transfer	201
14.4 Register Description.....	201
14.4.1 Register List	201
14.4.2 Channel Configuration Registers.....	202
14.4.3 SPI Configuration Registers.....	206

14.4.4 CLOCK CONFIGURATION REGISTERS.....	209
14.4.5 Mode Configuration Register	210
14.4.6 Interrupt Control Register	211
14.4.7 Interrupt Status Register	213
14.4.8 SPI Status Register.....	215
14.4.9 SPI Timeout Register.....	216
14.4.10 Data transmission register.....	216
14.4.11 Transfer Mode Register	217
14.4.12 Data Length Register	219
14.4.13 Data Receive Register	220
15 I2C Controller.....	221
15.1 Function overview.....	221
15.2 Main Features	221
15.3 Function description.....	221
15.3.1 Transmission rate selection.....	221
15.3.2 Interrupt and start-stop controllable.....	222
15.3.3 Fast output and detection signal	222
15.4 Register Description.....	222
15.4.1 Register List	222
15.4.2 Clock divider register_1	223
15.4.3 Clock divider register_2	223
15.4.4 Control Registers.....	224

15.4.5 Data Registers.....	.224
15.4.6 Transceiver Control Register.....	.225
15.4.7 TXR readout register.....	.227
15.4.8 CR read register.....	.227
16 I2S controller.....	.229
16.1 Function overview.....	.229
16.2 Main Features229
16.3 Function description.....	.229
16.3.1 Multiple Mode Support.....	229
16.3.2 Zero-crossing detection.....	.230
16.3.3 Efficient data transfer.....	.230
16.4 I2S/PCM Timing Diagram.....	.230
16.5 FIFO storage structure diagram.....	.232
16.6 I2S module working clock configuration.....	.234
16.7 Other function description:237
16.7.1 Zero-Crossing Detection:237
16.7.2 Mute function.....	.238
16.7.3 Interrupts.....	.238
16.7.4 FIFO Status Query.....	.238
16.8 Data transfer process.....	.239
16.8.1 The master sends audio data.....	.239
16.8.2 Slave receiving audio data.....	.239

16.8.3 The master receives audio data.....	240
16.8.4 Sending Audio Data from the Slave	241
16.8.5 Full-duplex mode.....	241
16.9 Register Descriptions.....	242
16.9.1 Register List	242
16.9.2 Control Registers.....	243
16.9.3 Interrupt Mask Register	248
16.9.4 Interrupt Flag Register	250
16.9.5 Status Register	254
16.9.6 Data transmission register.....	255
16.9.7 Data Receive Register	255
17 UART module.....	256
17.1 Function overview.....	256
17.2 Main Features	256
17.3 Function description.....	256
17.3.1 UART baud rate.....	256
17.3.2 UART data format.....	257
17.3.3 UART hardware flow control.....	259
17.3.4 UART DMA transfer.....	259
17.3.5 UART Interrupt.....	260
17.4 Register Descriptions.....	260
17.4.1 Register List	260

17.4.2 Data Flow Control Register	261
17.4.3 Automatic Hardware Flow Control Register	262
17.4.4 DMA setup register.....	263
17.4.5 FIFO Control Register	264
17.4.6 Baud Rate Control Register	265
17.4.7 Interrupt Mask Register.....	265
17.4.8 Interrupt Status Register	266
17.4.9 FIFO Status Register.....	268
17.4.10 TX start address register.....	268
17.4.11 RX Start Address Register.....	269
18 UART&7816 module.....	270
18.1 Function overview.....	270
18.2 Main Features	270
18.3 UART function description.....	271
18.4 7816 Functional Description.....	271
18.4.1 Introduction to the 7816.....	271
18.4.2 7816 interface.....	271
18.4.3 7816 Configuration.....	272
18.4.4 7816 Clock Configuration.....	272
18.4.5 7816 rate setting.....	273
18.4.6 7816 Power-On Reset.....	274
18.4.7 7816 warm reset.....	275

18.4.8 7816 Inactivation process	275
18.4.9 7816 data transfer.....	276
18.4.10 UART&7816 DMA transfer.....	276
18.4.11 UART&7816 Interrupt.....	277
18.5 Register Descriptions.....	277
18.5.1 Register List	277
18.5.2 Data Flow Control Register	278
18.5.3 Automatic Hardware Flow Control Register	281
18.5.4 DMA setup register.....	282
18.5.5 FIFO Control Register	283
18.5.6 Baud Rate Control Register	284
18.5.7 Interrupt Mask Register.....	285
18.5.8 Interrupt Status Register	285
18.5.9 FIFO Status Register.....	287
18.5.10 TX Start Address Register.....	288
18.5.11RX Start Address Register.....	288
18.5.12 7816 Guard Time Register	289
18.5.13 7816 Timeout time register.....	289
19 Timer module.....	290
19.1 Function overview.....	290
19.2 Main Features	290
19.3 Function description.....	290

19.3.1 Timing function.....	291
19.3.2 Delay function.....	291
19.4 Register Descriptions.....	291
19.4.1 Register List	291
19.4.2 Standard us configuration registers.....	292
19.4.3 Timer Control Register	292
19.4.4 Timer 1 Timing Value Configuration Register.....	294
19.4.5 Timer 2 Timing Value Configuration Register.....	294
19.4.6 Timer 3 Timing Value Configuration Register.....	294
19.4.7 Timer 4 Timing Value Configuration Register.....	294
19.4.8 Timer 5 Timing Value Configuration Register.....	295
19.4.9 Timer 6 Timing Value Configuration Register.....	295
19.4.10 Timer 1 current count value register.....	295
19.4.11 Timer 2 current count value register.....	295
19.4.12 Timer 3 current count value register.....	295
19.4.13 Timer 4 current count value register.....	296
19.4.14 Timer 5 current count value register.....	296
19.4.15 Timer 6 current count value register.....	296
20 Power Management Modules.....	298
20.1 Function overview.....	298
20.2 Main Features	298
20.3 Function description.....	298

20.3.1 Full-chip power control.....	298
20.3.2 Low Power Mode.....	299
20.3.3 Wake-up Mode.....	299
20.3.4 Timer0 timer.....	300
20.3.5 Real-time clock function.....	300
20.3.6 32K clock source switching and calibration.....	300
20.4 Register Description.....	301
20.4.1 Register List	301
20.4.2 PMU Control Registers.....	301
20.4.3 PMU Timer 0	304
20.4.4 PMU Interrupt Source Register.....	305
21 Real Time Clock Module	307
21.1 Function overview.....	307
21.2 Main Features	307
21.3 Function description.....	307
21.3.1 Timing function.....	307
21.3.2 Timing function.....	308
21.4 Register Description.....	308
21.4.1 Register List	308
21.4.2 RTC Configuration Register 1	308
21.4.3 RTC Configuration Register 2	309
22 Watchdog module.....	310

22.1 Function overview.....	310
22.2 Main Features	310
22.3 Function description.....	310
22.3.1 Timing function.....	310
22.3.2 Reset function.....	310
22.4 Register Description.....	311
22.4.1 Register List	311
22.4.2 WDG Timing Value Load Register.....	311
22.4.3 WDG current value register.....	312
22.4.4 WDG Control Register	312
22.4.5 WDG Interrupt Clear Register	312
22.4.6 WDG Interrupt Source Register.....	313
22.4.7 WDG Interrupt Status Register.....	313
23 PWM Controller	314
23.1 Function overview.....	314
23.2 Main Features	314
23.3 Function description.....	315
23.3.1 Input Signal Capture	315
23.3.2 DMA transfer captures.....	315
23.3.3 Support for single-shot and automount modes.....	315
23.3.4 Multiple Output Modes.....	315
23.4 Register Description.....	316

23.4.1 PWM register list.....	316
23.4.2 Clock divider register_01	317
23.4.3 Clock divider register_23	317
23.4.4 Control Registers.....	318
23.4.5 Period Register	321
23.4.6 Cycle count register.....	323
23.4.7 Compare Register	323
23.4.8 Dead Time Control Register.....	325
23.4.9 Interrupt Control Register	326
23.4.10 Interrupt Status Register	327
23.4.11 Channel 0 capture register.....	329
23.4.12 Brake Control Register	329
23.4.13 Clock divider register_4.....	330
23.4.14 Channel 4 Control Register_1	331
23.4.15 Channel 4 Capture Register	333
23.4.16 Channel 4 Control Register_2	334
24 QFLASH controller.....	338
24.1 Function overview.....	338
24.2 Main Features	338
24.3 Function description.....	338
24.3.1 Bus access.....	338
24.3.2 Register Access.....	338

24.3.3 Command configuration and startup.....	338
24.4 Register Descriptions.....	341
24.4.1 Register List	341
24.4.2 Command Information Register	341
24.4.3 Command Start Register	342
24.5 Common Commands of QFLASH.....	343
25 PSRAM interface controller.....	345
25.1 Function overview.....	345
25.2 Main Features	345
25.3 Function description.....	345
25.3.1 Pin Description	345
25.3.2 Access Mode Settings	346
25.3.3 PSRAM initialization.....	346
25.3.4 Access method of PSRAM	347
25.3.5 BURST function	347
25.4 Register Descriptions.....	348
25.4.1 Register List	348
25.4.2 Command Information Register	348
25.4.3 Timeout Control Register	349
26 Touch Sensor	365
26.1 Overview of module functions.....	365
26.2 Function Instructions.....	365

26.2.1 Basic Workflow.....	366
26.3 Register List:	366
26.3.1 Touch Sensor Control Register.....	367
26.3.2 Single-channel control register of touch button.....	368
26.3.3 Interrupt Control Register	368
27 W800 Security Architecture Design	370
27.1 Function overview.....	370
27.1.1 SRAM Secure Access Controller (SASC)	370
27.1.2 Trusted IP Controller (TIPC)	371
27.2 Security Architecture Block Diagram	371
27.3 Register Description.....	371
27.3.1 SASC register list.....	372
27.3.2 TIPC register.....	380
27.4 Instructions for use.....	382
27.4.1 Memory Safe Access (SASC).....	382
27.4.2 Trusted Access of Peripherals.....	385
28 Appendix 1. Chip Pin Definition.....	386
28.1 Chip Pinout.....	386
28.2 Chip pin multiplexing relationship.....	388
statement.....	390

Figure Catalog

Figure 1 W800 chip structure.....	41
Figure 2 W800 bus structure.....	43
Figure 3 W800 Clock Structure	47
Figure 5. System clock frequency division relationship.....	57
Figure 6 Host computer SPI send and receive data format.....	131
Figure 7 HSPI register read operation (big endian mode).....	131
Figure 8 HSPI register write operation (big endian mode).....	132
Figure 9 Register read operation (little endian mode).....	132
Figure 10 Register write operation (little endian mode).....	132
Figure 11 Port read operation (big endian mode).....	132
Figure 12 Port write operation (big endian mode).....	133
Figure 13 Port read operation (little endian mode).....	133
Figure 14 Port write operation (little endian mode).....	133
Figure 15 CPOL=0, CPHA=0	134
Figure 16 CPOL=0, CPHA=1	134
Figure 17 CPOL=1, CPHA=0.....	135
Figure 18 CPOL=1, CPHA=1.....	135
Figure 19 Main SPI processing interrupt flow.....	136
Figure 20 Flow chart of downlink data.....	137
Figure 21 Flowchart of downlink command.....	138
Figure 22 Upstream data (command) flowchart.....	139

Figure 23 SDIO internal storage map.....	143
Figure 24 CCCR register storage structure.....	144
Figure 25 FBR1 register structure.....	145
Figure 26 CIS storage space structure.....	145
Figure 27 SDIO Receive BD Descriptor.....	169
Figure 28 SDIO send BD descriptor.....	170
Figure 29 UART data length	257
Figure 30 UART stop bits	258
Figure 31 UART parity bit.....	258
Figure 32 UART hardware flow control connection.....	259
Figure 33 7816 Connection Diagram	272
Figure 34 7816 power-on reset sequence	274
Figure 35 7816 Warm Reset	275
Figure 36 7816 inactivation process.....	275
Fig. 37 7816 data transfer	276
Figure 38 W800 chip pinout.....	386
 table directory	
Table 1 List of AHB-1 bus masters.....	44
Table 2 List of AHB-1 bus slave devices.....	44
Table 3 List of AHB-2 bus masters.....	45
Table 4 AHB-2 bus slave device list.....	46
table 5 Detailed division of the bus device address space.....	49

Table 6 Startup Configurations.....	54
Table 8 Clock Reset Module Register List	60
Table 9 Software Clock Gating Enable Register	60
Table 10 Software Clock Mask Register	64
Table 11 Software Reset Control Register	65
Table 12 Clock Divide Configuration Registers.....	71
Table 13 Clock Select Register	73
Table 14 I2S clock control register.....	74
Table 14 Reset Status Register	76
Table 15 DMA address assignments.....	78
Table 16 DMA register list	80
Table 17 DMA Interrupt Mask Register	82
Table 18 DMA Interrupt Status Register	83
Table 19 UART selection register.....	84
Table 20 DMA Source Address Register	85
Table 21 DMA Destination Address Register	85
Table 22 DMA loop source start address register.....	85
Table 23 DMA loop destination start address register.....	86
Table 24 DMA Cycle Length Register	86
Table 25 DMA Channel Control Registers.....	86
Table 26 DMA Mode Select Register	87
Table 27 DMA data flow control registers.....	88

Table 28 DMA Transfer Bytes Register.....	90
Table 29 DMA linked list entry address register.....	90
Table 30 DMA current destination address register.....	90
Table 31 List of Cryptographic Module Registers.....	95
Table 32 Cryptographic Module Configuration Registers.....	96
Table 33 TRNG module control registers.....	99
Table 33 Cryptographic Module Control Registers.....	100
Table 34 Cryptographic Module Status Register	101
Table 35 RSA register list	102
Table 36 RSA Data X Register	103
Table 37 RSA Data Y Register	103
Table 38 RSA Data M Registers.....	103
Table 39 RSA Data D Register	104
Table 40 RSA Control Registers.....	104
Table 41 RSA parameter MC register.....	105
Table 42 RSA parameter N register.....	105
Table 43 GPIOA register list.....	107
Table 44 GPIOB register list.....	108
Table 45 GPIOA data register.....	109
Table 46 GPIOB data register.....	109
Table 47 GPIOA data enable register.....	110
Table 48 GPIOB data enable register.....	110

Table 49 GPIOA direction control register.....	110
Table 50 GPIOB direction control register.....	111
Table 51 GPIOA pull-up control register.....	111
Table 52 GPIOB pull-up and pull-down control registers.....	112
Table 53 GPIOA multiplexing selection register.....	112
Table 54 GPIOB multiplexing selection register.....	113
Table 55 GPIOA multiplexing selection register 1	114
Table 56 GPIOB multiplexing selection register 1	114
Table 57 GPIOA multiplexing selection register 0	114
Table 58 GPIOB multiplexing selection register 0	115
Table 59 GPIOA interrupt trigger mode configuration register.....	115
Table 60 GPIOB interrupt trigger mode configuration register.....	115
Table 61 GPIOA interrupt edge-triggered mode configuration register.....	116
Table 62 GPIOB interrupt edge-triggered mode configuration register.....	116
Table 63 GPIOA interrupt upper and lower edge-triggered configuration registers.....	116
Table 64 GPIOB interrupt upper and lower edge-triggered configuration registers.....	117
Table 65 GPIOA Interrupt Enable Configuration Register.....	117
Table 66 GPIOB interrupt enable configuration register.....	117
Table 67 GPIOA Raw Interrupt Status Register.....	118
Table 68 GPIOB Raw Interrupt Status Register.....	118
Table 69 GPIOA Masked Interrupt Status Register.....	118
Table 70 GPIOB Masked Interrupt Status Register.....	118

Table 71 GPIOA Interrupt Clear Control Register.....	119
Table 72 GPIOB Interrupt Clear Control Register.....	119
Table 73 HSPI Internal Access Registers.....	121
Table 74 HSPI FIFO clear register.....	122
Table 75 HSPI configuration registers.....	123
Table 76 HSPI Mode Configuration Registers.....	123
Table 77 HSPI Interrupt Configuration Registers.....	124
Table 78 HSPI Interrupt Status Register	124
Table 79 HSPI data upload length register.....	125
Table 80 HSPI interface configuration registers (master access)	125
Table 81 HSPI get data length register.....	127
Table 82 HSPI send data flag register.....	127
Table 83 HSPI Interrupt Configuration Register	128
Table 84 HSPI Interrupt Status Register	128
Table 85 HSPI data port 0.....	128
Table 86 HSPI Data Port 1.....	129
Table 87 HSPI command port 0.....	129
Table 88 HSPI Command Port 1.....	130
Table 89 SDIO CCCR register and FBR1 register list.....	145
Table 90 SDIO Fn1 address mapping relationship.....	157
Table 91 SDIO Fn1 part of the register (for HOST access)	158
Table 92 SDIO AHB bus registers.....	160

Table 93 WRAPPER Controller Registers.....	170
Table 94 WRAPPER Interrupt Status Register.....	172
Table 95 WRAPPER INTERRUPT CONFIGURATION REGISTER	172
Table 96 WRAPPER Upstream Command Ready Register.....	172
Table 97 WRAPPER downlink command buf ready register.....	173
Table 98 SDIO TX link enable register.....	173
Table 99 SDIO TX link address register.....	174
Table 100 SDIO TX enable register.....	174
Table 101 SDIO TX Status Register.....	174
Table 102 SDIO RX link enable register.....	175
Table 103 SDIO RX link address register.....	175
Table 104 SDIO RX enable register.....	176
Table 105 SDIO RX Status Register	176
Table 106 WRAPPER CMD BUF Base Address Register.....	177
Table 107 WRAPPER CMD BUF SIZE register.....	177
Table 108 SPI register list.....	201
Table 109 SPI channel configuration registers.....	202
Table 110 SPI configuration registers.....	206
Table 111 SPI Clock Configuration Registers.....	209
Table 112 SPI Mode Configuration Registers.....	210
Table 113 SPI Interrupt Control Registers.....	211
Table 114 SPI Interrupt Status Register.....	213

Table 115 SPI Status Register	215
Table 116 SPI Timeout Register	216
Table 117 SPI data transmission registers.....	216
Table 118 SPI transfer mode register.....	217
Table 119 SPI Data Length Register	219
Table 120 SPI Data Receive Registers.....	220
Table 121 I2C register list.....	222
Table 122 I2C clock divider register_1.....	223
Table 123 I2C clock divider register_2.....	223
Table 124 I2C Control Registers.....	224
Table 125 I2C Data Registers.....	224
Table 126 I2C Transceiver Control Register.....	225
Table 127 I2C TXR readout register.....	227
Table 128 I2C CR readout register.....	227
Table 129 I2S register list.....	242
Table 130 I2S Control Registers.....	243
Table 131 I2S Interrupt Mask Register.....	248
Table 132 I2S Interrupt Flag Register	250
Table 133 I2S Status Register.....	254
Table 134 I2S data transmission register.....	255
Table 135 I2S data receive register.....	255
Table 136 UART register list.....	260

Table 137 UART data flow control registers.....	261
Table 138 UART Auto Hardware Flow Control Registers.....	262
Table 139 UART DMA Setup Registers.....	263
Table 140 UART FIFO Control Registers.....	264
Table 141 UART Baud Rate Control Register.....	265
Table 142 UART Interrupt Mask Register	265
Table 143 UART Interrupt Status Register.....	266
Table 144 UART FIFO Status Register	268
Table 145 UART TX start address register.....	268
Table 146 UART RX start address register.....	269
Table 147 7816 Rate Settings.....	273
Table 148 UART&7816 register list.....	277
Table 149 UART&7816 data flow control register.....	278
Table 150 UART&7816 Automatic Hardware Flow Control Register.....	281
Table 151 UART&7816 DMA setting register.....	282
Table 152 UART&7816 FIFO Control Register.....	283
Table 153 UART&7816 Baud Rate Control Register.....	284
Table 154 UART&7816 Interrupt Mask Register.....	285
Table 155 UART&7816 Interrupt Status Register.....	285
Table 156 UART&7816 FIFO Status Register.....	287
Table 157 UART&7816 TX start address register.....	288
Table 158 UART&7816 RX start address register.....	288

Table 159 7816 Guard Time Register	289
Table 160 7816 Timeout Time Register	289
Table 161 Timer register list.....	291
Table 162 Timer standard us configuration register.....	292
Table 163 Timer Timer Control Register.....	292
Table 164 Timer 1 Timing Value Configuration Register.....	294
Table 165 Timer 2 Timing Value Configuration Registers.....	294
Table 166 Timer 3 Timing Value Configuration Registers.....	294
Table 167 Timer 4 Timing Value Configuration Registers.....	294
Table 168 Timer 5 Timing Value Configuration Registers.....	295
Table 169 Timer 6 Timing Value Configuration Registers.....	295
Table 170 PMU register list.....	301
Table 171 PMU Control Registers.....	301
Table 172 PMU Timer 0 Registers.....	304
Table 173 PMU Interrupt Source Registers.....	305
Table 174 RTC register list.....	308
Table 175 RTC Configuration Register 1	308
Table 176 RTC Configuration Register 2.....	309
Table 177 List of WDG Registers.....	311
Table 178 WDG Timing Value Load Register.....	311
Table 179 WDG current value register.....	312
Table 180 WDG Control Registers.....	312

Table 181 WDG Interrupt Clear Register.....	312
Table 182 WDG Interrupt Source Register	313
Table 183 WDG Interrupt Status Register.....	313
Table 184 PWM register list.....	316
Table 185 PWM clock divider register_01.....	317
Table 186 PWM clock divider register_23.....	317
Table 187 PWM Control Registers.....	318
Table 188 PWM Period Register	321
Table 189 PWM period number register.....	323
Table 190 PWM Compare Registers.....	323
Table 191 PWM Dead-Time Control Registers.....	325
Table 192 PWM Interrupt Control Register	326
Table 193 PWM Interrupt Status Register.....	327
Table 194 PWM Channel 0 Capture Register.....	329
Table 195 PWM Brake Control Register.....	329
Table 196 PWM clock divider register_4.....	330
Table 197 PWM Channel 4 Control Register_1	331
Table 198 PWM channel 4 capture register.....	333
Table 199 PWM Channel 4 Control Register_2	334
Table 200 QFLASH Controller Register List.....	341
Table 201 QFLASH command information register.....	341
Table 202 QFLASH command start register.....	342

Table 203 QFALSH common commands.....	343
Table 200 PSRAM controller register list.....	348
Table 201 PSRAM Control Setting Registers.....	348
Table 201 CS Timeout Control Register	349
Table 200 Touch Sensor Controller Register List.....	366
Table 201 Touch Sensor Control Setting Registers.....	367
Table 201 Touch key single channel setting register.....	368
Table 201 Touch key interrupt control register.....	368
Table 204 Chip pin multiplexing relationship.....	388

1 Introduction

1.1 Purpose of writing

The W800 chip is an embedded Wi-Fi SoC chip launched by Lianshengde Microelectronics. The chip is highly integrated, requires less peripheral devices, and is cost-effective

high. It is suitable for various smart products in the field of IoT (smart home). Highly integrated Wi-Fi and Bluetooth 4.2 Combo functions are its main features;

In addition, the chip integrates XT804 core, built-in QFlash, SDIO, SPI, UART, GPIO, I²C, PWM, I^S, 7816, LCD,

Interfaces such as Touch Sensor, support a variety of hardware encryption and decryption algorithms. In addition, the chip MCU contains a security kernel that supports code security permission settings.

The whole system supports firmware encrypted storage, firmware signature, security debugging, security upgrade and other security measures to improve product security features.

This document mainly describes the internal structure of the W800 chip, information on each functional module and detailed register usage information;

The main reference for the application. There are open source implementations of various functions in the SDK provided by Lianshengde Microelectronics, and developers can refer to the corresponding drivers

Programs, application examples to increase understanding of chip functions and register descriptions. There are no register descriptions for the Wi-Fi/BT part of this document.

1.2 References

For information on W800 chip package parameters, electrical characteristics, RF parameters, etc., please refer to "W800 Chip Product Specifications";

The W800 chip integrates the ROM program. The ROM program provides functions such as downloading firmware, MAC address reading and writing, and Wi-Fi parameter reading and writing.

For information, please refer to "WM_W800_ROM Function Brief";

The W800 chip has a built-in 2Mbytes QFlash memory, which is used as a storage space for codes and parameters. This document provides basic QFlash operations

for information. If there are requirements beyond the scope of this document, you need to refer to the QFlash manual;

W800 chip adopts Hangzhou Pingtou Ge XT804 core, 804 related function introduction, development materials, etc. can refer to Pingtou Ge company release information;

For more information, please refer to WinnerMicro's website (<http://www.winnermicro.com/>).

2 Features

ÿ Chip Packaging

ÿ QFN32 package, 4mm x 4mm.

ÿ Chip integration

ÿ Integrated XT804 processor, up to 240MHz

ÿ Integrated 288KB SRAM

ÿ Integrated 2MB FLASH

ÿ Integrated 8-channel DMA controller, supports 16 hardware applications, and supports software linked list management

ÿ Integrated PA/LNA/TR-Switch

ÿ Integrated 32.768KHz clock oscillator

ÿ Integrated voltage detection circuit

ÿ Integrated LDO

ÿ Integrated power-on reset circuit

ÿ Chip interface

ÿ Integrate 1 SDIO2.0 Device controller, support SDIO 1-bit/4-bit/SPI three operation modes, working clock range 0~50MHz

- ÿ Integrate 1 SDIO 2.0 HOST controller, support SDIO and SD card operation, working clock range 0~50MHz;
- ÿ Integrate a QSPI PSRAM interface, support a maximum capacity of 64MB PSRAM, and a maximum operating clock frequency of 80MHz;
- ÿ Integrate 5 UART interfaces, support RTS/CTS, baud rate range 1200bps~2Mbps
- ÿ Integrate a high-speed SPI slave interface, the operating clock range is 0~50MHz
- ÿ Integrate 1 SPI master/slave interface, the working clock of the master device is up to 20MHz, and the slave device supports up to 6Mbps data transfer rate
- ÿ Integrate an I2C controller, support 100/400Kbps rate
- ÿ Integrated PWM controller, support 5 channels of PWM
 - Single output or 2 PWM inputs. Maximum output frequency 20MHz, maximum input frequency 20MHz
- ÿ Integrated duplex I2S controller, support 32KHz to 192KHz I2S interface codec
- ÿ Integrate one 7816 interface, compatible with UART interface, support ISO-7816-3 T=0.T=1 mode; support EVM2000 Protocol
- ÿ Support a variety of hardware encryption and decryption modes, including RSA/AES/RC4/DES/3DES/RC4/SHA1/MD5/CRC8/CRC16/CRC32/TRNG
- ÿ Integrate one differential, or two single-ended 16bit ADC interfaces;
- ÿ Integrate 11-way Touch Sensor;
- ÿ Support up to 17 GPIO ports, each IO port has Rich reuse relationships. With input and output configuration options.
- ÿ WIFI protocol and function
 - ÿSupport [GB15629.11-2006, IEEE802.11 b/g/n;](#)
 - ÿ Support WMM/WMM-PS/WPA/WPA2/WPS
 - ÿSupport [WiFi Direct;](#)

- ÿ Support EDCA channel access mode;
- ÿ Support 20/40M bandwidth working mode;
- ÿ Support STBC, GreenField, Short-GI, support reverse transmission;
- ÿ Support RIFS frame interval;
- ÿ Support AMPDU, AMSDU;
- ÿ Support 802.11n MCS 0~7, MCS32 physical layer transmission rate gear, the transmission rate is up to 150Mbps;
- ÿ Support HT-immediate Compressed BlockAck, normal ACK, no ACK response mode;
- ÿ Support CTS to self;
- ÿ Support AP function; AP and STA are used at the same time;
- ÿ In the BSS network, multiple multicast networks are supported, and each multicast network supports different encryption methods.
 - 32 multicast networks and network access STA encryption;
 - ÿ When the BSS network supports as an AP, the total number of supported sites and groups is 32;
 - ÿ Receive Sensitivity:
 - ÿ 20MHz MCS7@-71dBm@10%PER;
 - ÿ 40MHz MCS7@-67dBm@10%PER;
 - ÿ 54Mbps@-73dBm@10%PER;
 - ÿ 11Mbps@-86dBm@8%PER;
 - ÿ 1Mbps@-96dBm@8%PER;
 - ÿ Support a variety of different received frame filtering options;
 - ÿ Support monitoring function;
- ÿ Bluetooth protocol and function

ÿ Integrated Bluetooth baseband processor/coprocessor, support BT/BLE4.2 protocol

ÿ Support various rates of DR/EDR;

ÿ Support BLE 1Mbps rate;

ÿ Power supply and power consumption

ÿ 3.3V single power supply;

ÿ Support Wi-Fi power saving mode power management;

ÿ Support work, sleep, standby, shutdown working modes;

ÿ Standby power consumption is less than 15uA;

3 Overview

This chip is a SOC chip that supports multi-interface and multi-protocol wireless local area network 802.11n (1T1R). The SOC chip integrates

RF Transceiver, CMOS PA Power Amplifier, Baseband Processor/Media Access Control, SDIO, SPI,

Low-power WLAN chip with interfaces such as UART and GPIO.

W800 chip supports GB15629.11-2006, IEEE802.11 b/g/n protocol, and supports STBC, Green Field,

Short-GI, Reverse Transmission, RIFS Interframe Interval, AMPDU, AMSDU, T-immediate Compressed Block Ack,

Rich protocols and operations such as normal ACK, no ACK, and CTS to self.

The W800 chip integrates the RF transceiver front-end, A/D and D/A converters. It supports DSSS (Direct Sequence Spread Spectrum) as well as OFDM

(Orthogonal Frequency Division Multiplexing) modulation mode, with data descrambling capability, supports a variety of different data transmission rates. in the analog front end of the transceiver

The equipped transceiver AGC function enables the system-on-a-chip to obtain the best performance. The W800 chip also includes a built-in enhanced signal monitor,

The influence of multipath effect can be largely eliminated.

In terms of security, the W800 chip not only supports the national standard WAPI encryption, but also supports the international standard WEP, TKIP, CCMP encryption,

These hardware components enable data transmission systems based on the chip to obtain data similar to those of non-encrypted communications during secure communications.

data transmission performance.

In addition to supporting the energy-saving operations specified by IEEE802.11 and Wi-Fi protocols, the W800 chip also supports user-customized energy-saving solutions. chip

It supports four working modes: work, sleep, standby and shutdown, so that the entire system can achieve low power consumption, and it is convenient for users to adapt to their own use fields.

different scenarios for energy saving.

The W800 chip integrates a high-performance 32-bit embedded processor, a large number of memory resources, and a wealth of peripheral interfaces, which are convenient for users.

It is easy to apply the chip to the secondary development of a specific product.

The W800 chip supports AP function, which can realize the establishment of 5 SSID networks at the same time, and realize the function of 5 independent APs. Support for creating multiple

Multicast network function. It can realize the function of establishing a BSS network as an AP while joining other networks as a STA.

The W800 chip supports the WPS method, allowing users to implement an encrypted complete network with one-click operation to ensure the security of information sex.

The multi-function and high integration of W800 chip ensures that the WLAN system does not need too many off-chip circuits and external memory.

4 Chip Structure

4.1 Chip structure

The following figure describes the overall structure of the W800 chip, the core part includes XT804 CPU, 288KB SRAM and 20KB ROM storage space.

As the constant power supply module of the chip, the PMU part provides power-on sequence management, start-up clock, and real-time clock functions. Provides a wealth of peripherals function and hardware encryption and decryption functions. The Wi-Fi part integrates MAC, BB and RF.

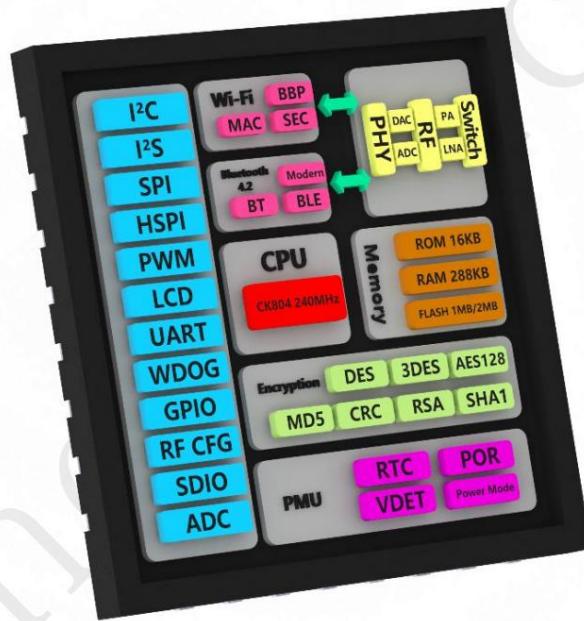


Figure 1 W800 chip structure diagram

Winner Micro

4.2 Bus Structure

The W800 chip consists of a two-level bus, as shown in the figure below

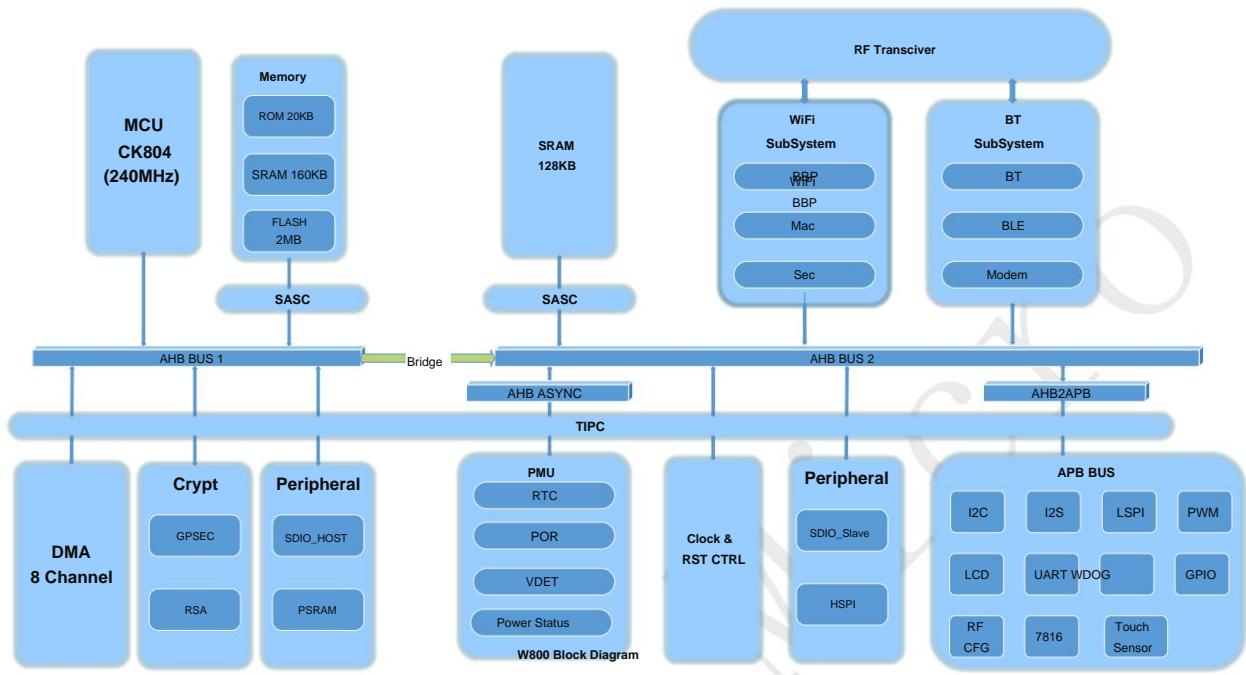


Figure 2 W800 bus structure diagram

(1) AHB-1 bus

This bus has four masters - XT804, DMA, GPSEC and 5 slaves.

XT804 is a 32-bit energy-efficient embedded CPU core oriented to the control field. It adopts 16/32-bit mixed coding instruction system.

A streamlined and efficient 3-stage pipeline.

The XT804 provides a variety of configurable functions, including hardware floating point unit, on-chip cache, DSP acceleration unit, trusted protection technology,

On-chip tightly coupled IP, etc., users can configure according to application needs. In addition, XT804 provides multi-bus interface, supports system bus, finger

Flexible configuration of the order bus and data bus. XT804 has made special acceleration for interrupt response, and the interrupt response delay is only 13 cycles.

The bus clock operates at the fastest frequency of 240MHz and can be configured to 240/160/120/80/40MHz, or lower.

Table 1 AHB-1 bus master list

main device	Function
CPU	Complete chip register configuration, memory management and use, and complete 802.11MAC protocol. Highest Operating frequency 240MHz
DMA	DMA supports an independent 8-channel DMA module with a linked list structure, and supports 16 on-chip hardware DMA request sources.
GPSEC	Universal encryption module, supports DES/3DESSHA1/AES/MD5/RC4/CRC/PRDN. automatic completion Data blocks in the specified memory space are encrypted and written back.

Table 2 AHB-1 bus slave device list

Slave function	
ROM	ROM is used to store the initialization firmware after the CPU is powered on. It mainly completes the initial configuration of the chip register space and other work. After completing the above work, the CPU control The control is given to the firmware stored in FLASH.
AHB2AHB	Complete the conversion of CPU bus clock domain to BusMatrix2 bus clock domain master access. Require The clock domains must be of the same source, and the ratio of the CPU clock to the BusMatrix2 clock frequency is M:1, M is an integer greater than or equal to 1.
FLASH	FLASH stores firmware code and operating parameters.
SRAM 160KB	SRAM 160KB can be used to store instructions or data, and firmware can use this memory as needed.
RSA	Supports RSA encryption and decryption operations up to 2048bit

GPSEC general encryption/decryption module, supports SHA1/AES/MD5/RC4/CRC/TRNG. autocomplete	Encrypt/decrypt and write back data blocks in a given memory space.
SDIO_HOST SDIO 2.0 standard SDIO HOST controller; SDIO interface peripherals can be accessed through this interface.	The SDIO interface clock is obtained by dividing the bus clock and supports up to 50MHz.
PSRAM_CTRL PSRAM controller for QSPI interface. An external PSRAM can be accessed through this controller. QSPI connection	The port clock is obtained by dividing the frequency of the bus clock, and it supports up to 80MHz clock.

(2) AHB-2 bus

This bus has 4 master devices and 3 slave devices. Using the crossbar connection structure, it can realize different master devices to different slave devices.

access at the same time, thereby increasing the bandwidth. The bus clock operates at the fastest frequency of 40MHz and can be configured to be lower as required.

Table 3 AHB-2 bus master list

main device	Function
MAC	802.11MAC control protocol processing module. Operations on the bus mainly include sending read data Data, receive write data, and send completion descriptor write-back operations.
SEC	The security module completes the encryption, decryption and movement of the sent and received data. When sending, the data and The MAC descriptor is moved to the specified location and encryption is completed; when receiving, the received data and The MAC receives the descriptor and moves to the specified location and completes the decryption.
AHB2AHB	Transition of bus master access from the AHB-1 bus to the AHB-2 bus.
SDIO/HSPI	Connect the host to the chip through the SDIO2.0 device controller or the high-speed SPI slave device controller. Accesses translate to AHB bus signals and access content memory and register space.

Each master device adopts a fixed priority, and the priority decreases from top to bottom.

Table 4 AHB-2 bus slave device list

slave device	Function
SRAM 128KB	Used to store upstream and downstream data buffer, SDIO/SPI/UART interface uses this RAM as data cache
	Configuration register configuration space, high-speed module configuration registers are uniformly addressed here.
	All low-speed modules of APB access the space, and various low-speed modules are connected using APB bus.
BT_CORE Bluetooth controller.	

4.3 Clock Structure

W800 uses 24/40MHz crystal as SoC clock source, built-in 1 DPLL output 480MHz, supply

Used by CPU, system bus, data bus and WiFi system; on-chip additional built-in 32.768KHZ RC oscillator for PMU

and LCD module use. An overview of the clock structure is shown in the figure below.

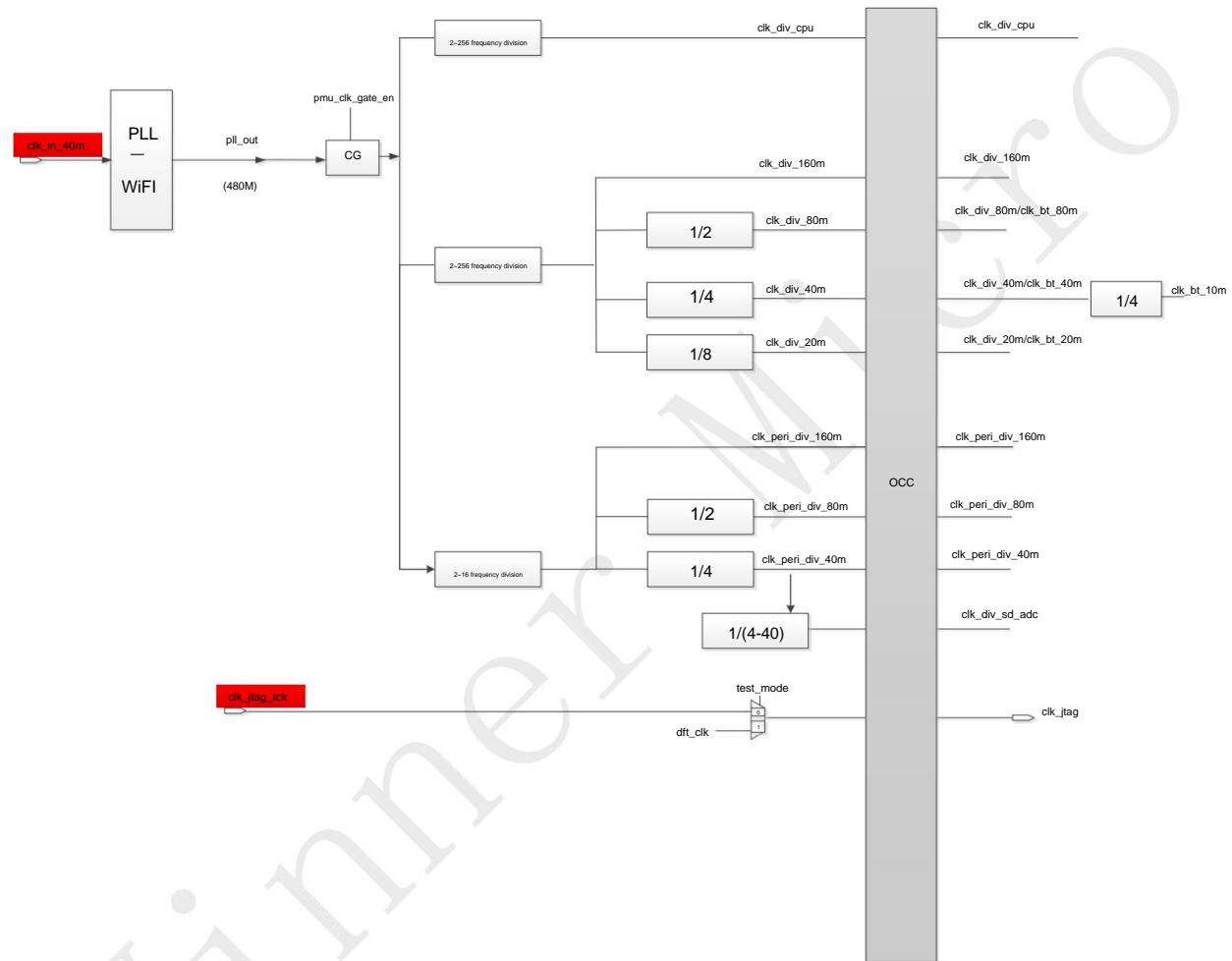
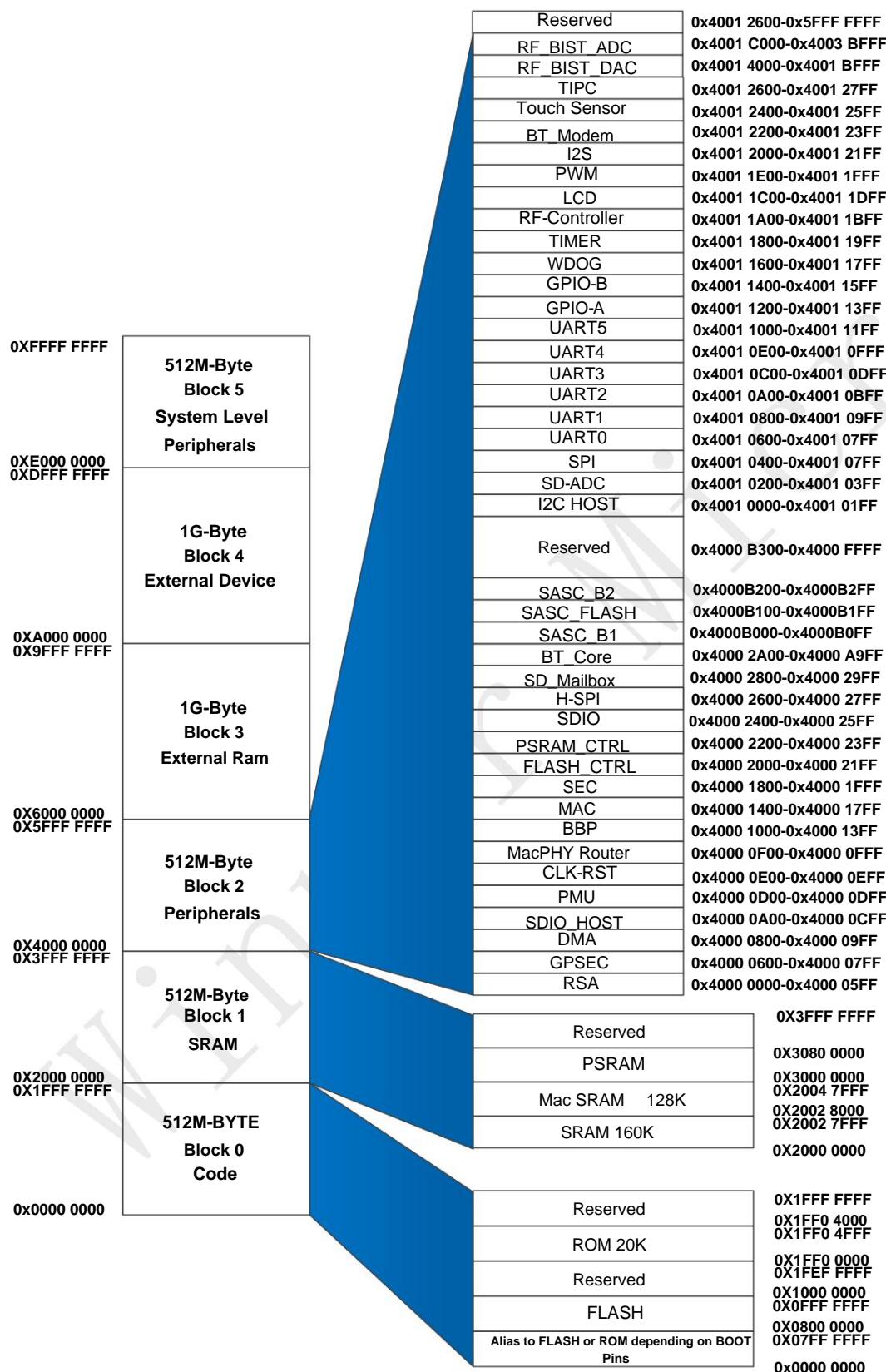


Figure 3 W800 clock structure

4.4 Address space



W800 Address Mapping

XT804 supports 4G storage space, which is divided into 6 blocks as shown in the figure above, which are code area, memory area, on-chip peripherals, and off-chip storage.

area, off-chip peripherals and system peripherals area. According to requirements, the on-chip storage space of w800 is mapped to the first three areas as shown in Figure 3.

table 5 Detailed division of bus device address space

bus from equipment	BootMode=0	Address space breakdown	Remark
ROM	0x0000 0000 ~ 0x0004 FFFF		Store the solidified firmware code
FLASH	0x0800 0000 ~ 0x0FFF FFFF		Stored as a dedicated instruction device.
SRAM	0x2000 0000 ~ 0x2002 7FFF		Firmware memory and instruction storage Area
Mac RAM	0x2002 8000 ~ 0x2004 7FFF		SDIO/H-SPI/UART data cache
PSRAM	0x3000 000~0x30800000		Peripheral memory
CONFIG	0x4000 0000 ~ 0x4000 2FFF	0x4000 0000 ~ 0x4000 05FF	RSA configuration space
		0x4000 0600 ~ 0x4000 07FF	GPSEC configuration space
		0x4000 0800 ~ 0x4000 09FF	DMA configuration space
		0x4000 0A00 ~ 0x4000 0CFF	SDIO_HOST configuration is empty
		0x4000 0D00 ~ 0x4000 0DFF	PMU configuration space

	0x4000 0E00 ~ 0x4000 0EFF	Clock and Reset configuration space
	0x4000 0F00 ~ 0x4000 0FFF	MacPHY Router configuration space
	0x4000 1000 ~ 0x4000 13FF	BBP configuration space
	0x4000 1400 ~ 0x4000 17FF	MAC configuration space
	0x4000 1800 ~ 0x4000 1FFF	SEC configuration space
	0x4000 2000 ~ 0x4000 21FF	FLASH Controller configuration space
	0x4000 2200 ~ 0x4000 23FF	PSRAM_CTRL configuration space
	0x4000 2400 ~ 0x4000 25FF	SDIO Slave configuration is empty between
	0x4000 2600 ~ 0x4000 27FF	H-SPI configuration space
	0x4000 2800 ~ 0x4000 29FF	SD Wrapper configuration space
	0x4000 2A00 ~ 0x4000 A9FF	BT Core configuration space
	0x4000 B000 ~ 0x4000 B0FF	SASC-B1 Level 1 Bus Memory Security Configuration Mode piece
	0x4000 B100 ~ 0x4000 B1FF	SASC-Flash Flash Security Configuration Module

		0x4000 B200 ~ 0x4000 B2FF	SASC-B2 Secondary bus memory security configuration module piece
APB	0x4001 0000 ~ 0x 4001 C000	0x4001 0000 ~ 0x4001 01FF	I2C master
		0x4001 0200 ~ 0x4001 03FF	Sigma ADC
		0x4001 0400 ~ 0x4001 07FF	SPI master
		0x4001 0600 ~ 0x4001 07FF	UART0
		0x4001 0800 ~ 0x4001 09FF	UART1
		0x4001 0A00 ~ 0x4001 0BFF	UART2
		0x4001 0C00 ~ 0x4001 0DFF	UART3
		0x4001 0E00 ~ 0x4001 0FFF	UART4
		0x4001 1000 ~ 0x4001 11FF	UART5
		0x4001 1200 ~ 0x4001 13FF	GPIO-A
		0x4001 1400 ~ 0x4001 15FF	GPIO-B
		0x4001 1600 ~ 0x4001 17FF	WatchDog
		0x4001 1800 ~ 0x4001 19FF	Timer
		0x4001 1A00 ~ 0x4001 1BFF	RF_Controller
		0x4001 1C00 ~ 0x4001 1DFF	LCD
		0x4001 1E00 ~ 0x4001 1FFF	PWM

	0x4001 2000 ~ 0x4001 22FF	I2S
	0x4001 2200 ~ 0x4001 23FF	BT-modem
	0x4001 2400 ~ 0x4001 25FF	Touch Sensor
	0x4001 2600 ~ 0x4001 25FF	TIPC Interface Security Settings
	0x4001 4000 ~ 0x4000 BFFF	RF_BIST DAC transmit Memory
	0x4001 C000 ~ 0x4003 BFFF	RF_BIST ADC receive Memory
	0x4001 3C00 ~ 0x5FFF FFFF	RSV

4.4.1 SRAM

W800 has built-in 288KB SRAM. Among them, 160KB is mounted on the first-level AHB bus, and 128KB is mounted on the second-level AHB bus. CPU

Devices on the first-level bus can access all memory areas, but devices on the second-level bus can only access 128KB of memory on the second-level bus.

4.4.2 Flash

4.4.2.1 QFlash

W800 integrates 2MBytes QFlash inside. The XIP method is implemented on the QFlash through the integrated 32KB cache inside the chip.

sequence. When the program is running, the CPU first reads the instructions from the Cache.

QFlash reads the instruction and stores it in the Cache. Therefore, when the continuous running code size is less than 32K, the CPU will not need to read from QFlash

Fetch instructions, at which time the CPU can run at a higher frequency. The above method is the operation mode of the read command, and the RO segment of the entire Image will be

Operate in this way. This process requires no user intervention.

QFlash can also store data. When the user program needs to read and write data in QFlash, it needs to use the built-in QFlash controller.

Operation, QFlash provides the corresponding address, instruction and other registers to assist the user to achieve the desired operation. For specific description, please refer to QFlash

The controller corresponds to the chapter.

Users need to pay attention that when the program reads or writes data, there is no need to perform state judgment, wait and other operations, because the QFlash control

The device itself will judge. When the QFlash controller returns, the read or write is complete.

4.4.2.2 SPI Flash

In addition to supporting 6PIN QFlash interface (built-in PIN, not packaged), W800 chip also supports low-speed SPI interface access. The SPI

The maximum operating frequency of the interface can reach 20MHz, and it supports the master-slave function. For a detailed description, please refer to the corresponding chapter of the SPI interface.

4.4.3 PSRAM

W800 has a built-in PSRAM controller with SPI/QSPI interface, supports external PSRAM device access with a maximum capacity of 64Mb, and provides bus

mode of PSRAM read, write and erase operations. The maximum read and write speed is 80MHz. When the storage capacity needs to be expanded, the off-chip PSRAM can be used to expand

Fill code storage space or data storage space. PSRAM also supports XIP execution of programs, and CPU Cache also supports cache

data in PSRAM.

4.5 Startup Configuration

After the W800 chip is powered on, the CPU will start to execute the firmware in the ROM and load the user image at the specified address in the Flash.

When the ROM firmware starts to run, it will read the BootMode (PA0) pin, and judge to enter the boot state according to the signal of the pin:

Table 6 Startup configuration

BootMode	start condition	boot mode
high		normal startup process
Low	Continuous <30ms, quick test mode is invalid	normal startup process
	Last >=30ms	Enter functional mode
Note:		
Test mode: chip test function, user cannot operate.		
Function mode: Enter the basic functions implemented by ROM, such as: downloading firmware, programming MAC address, etc. For details, please refer to		

"WM_W800_ROM Function Brief.pdf"

Typically, the BootMode pins should be used in production or debug stages. During the production phase, the user continuously pulls the BootMode pin

If it is low for more than 30ms, it enters the function mode, and can quickly burn the Flash.

In the scenario of product rework or repair, the chip does not enter the "highest security level" (for a description of the security level, please refer to

"WM_W800_ROM Function Brief"), you can enter the function mode through this pin, erase the old Image, write the new Image.

In the debugging stage, no matter what the firmware is faulty, you can enter the serial port by continuously pulling the BootMode pin down for more than 30ms

Download function, burn new firmware.

5 Clock and reset module

5.1 Function overview

The clock and reset module completes the software control of the chip clock and reset system. Clock control includes clock frequency conversion, clock shutdown and self-adaptation

should be gated; reset control includes soft reset control of the system and sub-modules.

5.2 Main Features

- ÿ Supports clock shutdown of each module
- ÿ Support some modules clock adaptive shutdown
- ÿ Support software reset of each module
- ÿ Support CPU frequency setting
- ÿ Support ADC/DAC loopback test
- ÿ Support I2S clock setting

5.3 Functional Description

5.3.1 Clock Gating

By configuring the clock gating enable register CLK_GATE_EN, the clock of the specified function can be controlled to turn off, so as to turn off the function of a certain module.

able purpose.

In order to provide the flexibility of firmware to control the power consumption of the system, the clock and reset module provides the clock gating function of each module in the system. when closed

When the clock of the corresponding module is stopped, the digital logic and clock tree of the module will stop working, which can reduce the dynamic power consumption of the system.

The switch of each module corresponds to the detailed description of the register SW_CLKG_EN.

5.3.2 Clock Adaptive Shutdown

The chip adaptively shuts down the clocks of certain functional modules according to the transition of certain internal states.

Users, please do not change the configuration, otherwise it may cause system abnormality when configuring the PMU function.

5.3.3 Function reset

The chip provides the soft reset function of each subsystem, and the subsystem reset can be achieved by setting the corresponding BIT of SW_RST_CTRL to 0.

However, the reset state will not be cleared automatically, and the corresponding BIT of SW_RST_CTRL needs to be set to 1 to resume normal operation.

The soft reset function does not reset the CPU and WatchDog.

In this register, the reset operation of APB/BUS1/BUS2 (corresponding to APB bus, system bus and data bus) is not recommended, which will cause the system

The system access device is abnormal.

5.3.4 Clock division

The W800 system uses 40MHz/24MHz crystal as the system clock source, the system has built-in DPLL, and the fixed output 480MHz clock is used as the clock source.

The clock source of the whole system (as shown in the figure below).

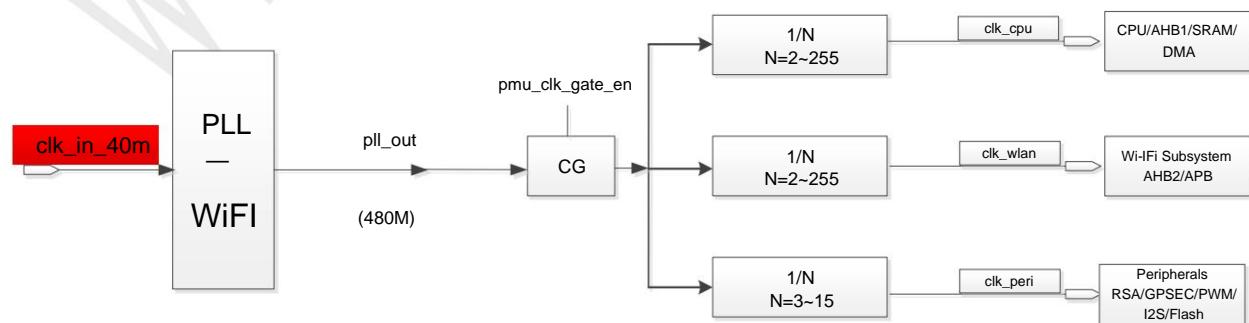


Figure 4 System clock frequency division relationship

The clock of the system bus is consistent with the CPU clock, and the clock of the data bus is fixed to 1/4 of the WLAN root clock.

The WLAN root clock is also the clock source of the entire WLAN system.

This module provides the function of setting CPU clock and WLAN root clock for firmware to adjust system performance and power consumption.

Set the BIT[7:0] of the SYS_CLK_DIV register to adjust the CPU clock frequency division factor. The source clock of the CPU clock frequency division is the DPLL output, fixed at 480MHz. The default value of the CPU clock frequency division factor is 6, that is, the default operating frequency of the CPU is 480MHz divided by 6.

That is 80MHz. This parameter can be reconfigured when the clock required by the CPU needs to be adjusted.

The CLK_PERI clock provides the root clock of the running clock of the encryption module in the SoC system, and the root clock of the running clock of some interfaces, more than

Such as PWM interface, I2S interface, Flash interface clock. This clock is also derived by dividing the 480MHz output from the DPLL. normal working condition

The lower frequency should be fixed to divide by 3 to get the CLK_PERI root clock of 160MHz. Divide by 2 and divide by 4 by CLK_PERI root clock

80MHz and 40MHz are provided for encryption module and interface module.

Set the BIT[15:8] of the SYS_CLK_DIV register to adjust the WLAN clock frequency division factor. The default frequency division factor is 3, that is, for DPLL

The 480MHz output frequency is divided by 3, and the 160MHz clock is obtained, which is sent to the WLAN as the root node clock (the WLAN continues to divide the frequency to obtain more

For the detailed low frequency clock used by the WLAN system.

Note: If you want the WLAN system to work normally, the WLAN root clock needs to be kept at 160MHz, otherwise the WLAN system will fail.

When the WLAN system is not required to work, the WLAN root clock can be lowered to reduce the dynamic power consumption of the system.

When changing the system clock configuration, it should be noted that the ratio of the system bus to the data bus needs to be maintained at M:1, where M is an integer,

The minimum is 1. When changing the system clock configuration, it is also necessary to update the BIT [23:16] of the register SYS_CLK_DIV at the same time, set the correct ratio example coefficient. Otherwise, accessing the data bus will result in abnormal data.

[15:8] of SYS_CLK_SEL provides the frequency division factor for setting the operating frequency of SAR_ADC, which is divided by 40M as the clock source. Frequency division

The number is the assigned frequency division value.

BIT[4] of SYS_CLK_SEL is to configure the clock frequency selection of the core operation of the RSA module, which can be 80MHz or 160MHz.

BIT[5] is to configure the clock frequency selection of the core operation of the GPSEC module, which can be 80MHz or 160MHz.

BIT[6] is to configure the clock frequency selection of the external bus of the FLASH module, which can be 40MHz or 80MHz.

When you need to reconfigure cpu_clk_divider, wlan_clk_divider, bus2_syncdn_factor, sdadc_fdiv, you need to set

Bit BIT[31] of SYS_CLK_DIV, the hardware automatically updates the above four parameters to the frequency divider, and then clears BIT[31].

I2S_CLK_CTRL provides the clock configuration function of the I2S module.

5.3.5 Debug function control

The user can enable and disable the JTAG function by setting the value of DEBUG_CTRL (SYS_CLK_SEL-BIT[16]).

5.4 Register Description

5.4.1 Register List

Table 7 Clock Reset Module Register List

offset address	name	abbreviation	access	describe	reset value
0X0000 Software	Clock Gating Enable Register SW_CLKG_EN		RW Whether	the software configuration module turns off the clock	0X0000_7FFF
0X0004 Software	clock mask register SW_CLK_MASK RW			Whether the software configuration module is adaptively shut down bell	0X0000_007E
0X0008 Reserved					
0X000C Software	reset control register SW_RST_CTRL		RW software	configuration reset module	0X01FF_FFFF
0X0010 Clock divider configuration register		SYS_CLK_DIV	RW configures	the clock divider ratio	0X0000_2212
0X0014 Debug Control Register		DEBUG_CTRL	RW Configure	ADC/DAC loopback test	0X0000_0000
0X0018	I2S Clock Control Register	I2S_CLK_CTRL	RW config	ure the I2S clock	0X0000_0000
0X001C Reset Status Register		RESET_STATUS RW		View CPU soft reset and Watchdog reset state	0x0000_0000

5.4.2 Software Clock Gating Enable Register

Table 8 Software Clock Gating Enable Register

bit access		Instructions	reset value
[31:22] RO		reserve	
[twenty one]	RW	soft_touch_gate_en Configure the gate control of the touch_sensor module clock. By default, the touch_sensor module gate control is enabled. 0: touch_sensor module clock is off	1'b1

		1: touch_sensor clock on	
[20]	RW	<p>soft_bt_gate_en</p> <p>Configure the gate control of the BT./BLE module clock. By default, the gate control of the BT/BLE module is enabled.</p> <p>0: BT/BLE module clock is off</p> <p>1: BT/BLE clock on</p>	1'b1
[19]	RW	<p>soft_qspi_gate_en</p> <p>Configure the gating of the clock of the qspi_ram module. By default, the gating of the qspi_ram module is enabled.</p> <p>0: qspi_ram module clock is off</p> <p>1: qspi_ram clock on</p>	1'b1
[18]	RW	<p>soft_sdio_m_gate_en</p> <p>Configure the gating of the clock of the sdio_master module. By default, the gating of the sdio_master module is enabled.</p> <p>0: sdio_master module clock is off</p> <p>1: sdio_master clock on</p>	1'b1
[17]	RW	<p>soft_gpsec_gate_en</p> <p>Configure gpsec module clock gating, gpsec module gating is enabled by default</p> <p>0: gpsec module clock is off</p> <p>1: gpsec clock on</p>	1'b1
[16]	RW	<p>soft_rsa_gate_en</p> <p>Configure the gating of the RSA clock, the default RSA gating is enabled</p> <p>0: RSA module clock is off</p> <p>1: RSA clock on</p>	1'b1
[15]	RW	soft_i2s_gate_en	1'b1

		Configure i2s clock gating, i2s gating is enabled by default 0: i2s clock is off 1: i2s clock on	
[14]	RW	soft_lcd_gate_en Configure the gating of the lcd clock, the default lcd gating is enabled 0: LCD clock off 1: LCD clock on	1'b1
[13]	RW	Soft_pwm_gate_en Configure the gating of the pwm clock, the default pwm gating is enabled 0: pwm clock off 1: pwm clock on	1'b1
[12]	RW	soft_sd_adc_gate_en Configure the gating of sd_adc_clock, the default sd_adc_gating is enabled 0: sd_adc_clock off 1: sd_adc_clock on	1'b1
[11]	RW	soft_gpio_gate_en Configure the gating of the GPIO clock, the default GPIO gating is enabled 0: GPIO clock off 1: GPIO clock on	1'b1
[10]	RW	soft_timer_gate_en Configure the gate control of the timer clock, the default timer gate control is enabled 0: timer clock off	1'b1

		1: timer clock on	
[9]	RW	<p>soft_rf_cfg_gate_en: used internally, do not modify</p> <p>Configure the gating of the rf_cfg clock, the default rf_cfg gating is enabled</p> <p>1'b0: rf_cfg clock off</p> <p>1'b1: rf_cfg clock on</p>	1'b1
[8]	RW	<p>soft_dma_gate_en</p> <p>Indicates whether the clock supplied to the dma clock domain is turned off</p> <p>1'b0: dma clock off</p> <p>1'b1: dma clock on</p>	1'b1
[7]	RW	<p>soft_ls_spi_gate_en</p> <p>Configure the gating of the low-speed spi clock, the default low-speed spi gating is enabled</p> <p>1'b0: low speed spi clock off</p> <p>1'b1: low speed spi clock on</p>	1'b1
[6]	RW	<p>soft_uart5_gate_en</p> <p>Configure the gate control of uart5, the default uart5 is enabled</p> <p>0: uart5 is off</p> <p>1: uart5 is on</p>	1'b1
[5]	RW	<p>soft_uart4_gate_en</p> <p>Configure the gate control of uart4, the default uart4 is open</p> <p>0: uart4 is off</p> <p>1: uart4 is on</p>	1'b1
[4]	RW	soft_uart3_gate_en	1'b1

		Configure the gate of uart3, the default uart3 is enabled 0: uart3 is off 1: uart3 is enabled	
[3]	RW	soft_uart2_gate_en Configure the gate control of uart2, the default uart2 is enabled 0: uart2 is off 1: uart2 is enabled	1'b1
[2]	RW	soft_uart1_gate_en Configure the gating of the uart1 clock, the default uart1 gating is enabled 1'b0: uart1 clock off 1'b1: uart1 clock on	1'b1
[1]	RW	soft_uart0_gate_en Configure the gating of the uart0 clock, the default uart0 gating is enabled 1'b0: uart0 clock off 1'b1: uart0 clock on	1'b1
[0]	RW	soft_i2c_gate_en Configure i2c clock gating, i2c gating is enabled by default 1'b0: i2c clock off 1'b1: i2c clock on	1'b1

5.4.3 Software Clock Mask Register

Table 9 Software Clock Mask Register

bit access		Instructions	reset value
[6]	RW	<p>soft_cpu_clk_gt_mask</p> <p>Indicates whether the clock supplied to the CPU clock domain (including CPU, bus1, ROM, SRAM) can be adaptive</p> <p>Shutdown (when the CPU needs to enter the WFI state, do not set the adaptive shutdown)</p> <p>1'b0: Allows adaptive turn-off and turn-on</p> <p>1'b1: Adaptive turn-off and turn-on are not allowed</p>	1'b1
[5 : 2]	RW	Reserved for internal use, do not modify	
[1]	RW	<p>soft_sdioahb_clk_gt_mask</p> <p>Indicates whether the clock supplied to the sdio ahb clock domain can be turned off adaptively</p> <p>1'b0: Allows adaptive turn-off and turn-on</p> <p>1'b1: Adaptive turn-off and turn-on are not allowed</p>	1'b1
[0]	RW	<p>soft_pmu_clk_gt_mask</p> <p>There is a gate control unit after the clock output by pll, which is configured by this register to indicate whether it is allowed to be turned off by the PMU.</p> <p>1'b0: Allows the PMU to shut down the gate unit, thereby shutting down the clock</p> <p>1'b1: PMU is not allowed to shut down the gate unit</p>	1'b0

5.4.4 Software Reset Control Register

Table 10 Software Reset Control Register

bit access		Instructions	reset value
[31]	RW	<p>soft_touch_RST_N</p> <p>Software reset touch_sensor module</p> <p>0: reset</p>	1'b1

		1: Reset release	
[30]	RW	<p>soft_rst_flash_n</p> <p>Software reset the flash controller module</p> <p>0: reset</p> <p>1: Reset release</p>	1'b1
[29]	RW	<p>soft_rst_bt_n</p> <p>Software reset BT module</p> <p>0: reset</p> <p>1: Reset release</p>	1'b1
[28]	RW	<p>soft_rst_qspi_ram_n</p> <p>Software reset qspi_ram module</p> <p>0: reset</p> <p>1: Reset release</p>	1'b1
[27]	RW	<p>soft_rst_sdio_m_n</p> <p>Software reset sdio_master module</p> <p>0: reset</p> <p>1: Reset release</p>	1'b1
[26]	RW	<p>soft_rst_gpsec_n</p> <p>software reset gpsec module</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	1'b1
[25]	RW	soft_rst_rsa_n	1'b1

		Software reset RSA module 1'b0: reset 1'b1: reset release	
[twenty four]	RW	soft_RST_I2S_N Software reset i2s module 1'b0: reset 1'b1: reset release	1'b1
[twenty three]	RW	soft_RST_LCD_N software reset lcd module 1'b0: reset 1'b1: reset release	1'b1
[twenty two]	RW	soft_RST_PWM_N software reset pwm module 1'b0: reset 1'b1: reset release	1'b1
[twenty one]	RW	soft_RST_SAR_ADC_N Software reset sar_adc module 1'b0: reset 1'b1: reset release	1'b1
[20]	RW	soft_RST_TIMER_N Software reset timer module 1'b0: reset	1'b1

		1'b1: reset release	
[19]	RW	soft_RST_GPIO_N software reset gpio module 1'b0: reset 1'b1: reset release	1'b1
[18]	RW	soft_RST_RF_CFG_N Software reset to configure the RF register module (for internal use, do not modify) 1'b0: reset 1'b1: reset release	1'b1
[17]	RW	soft_RST_SPI_S_N Software reset high-speed spi module 1'b0: reset 1'b1: reset release	1'b1
[16]	RW	soft_RST_SPI_M_N Software reset low speed spi module 1'b0: reset 1'b1: reset release	1'b1
[15]	RW	soft_RST_UART5_N Software reset on-chip uart5 module 1'b0: reset 1'b1: reset release	1'b1
[14]	RW	soft_RST_UART4_N	1'b1

		<p>Software reset on-chip uart4 module</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	
[13]	RW	<p>soft_RST_uart3_n</p> <p>Software reset on-chip uart3 module</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	1'b1
[12]	RW	<p>soft_RST_uart2_n</p> <p>Software reset on-chip uart2 module</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	1'b1
[11]	RW	<p>soft_RST_uart1_n</p> <p>Software reset on-chip uart1 module</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	1'b1
[10]	RW	<p>soft_RST_uart0_n</p> <p>Software reset on-chip uart0 module</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	1'b1
[9]	RW	<p>soft_RST_i2c_n</p> <p>Software reset on-chip i2c module</p> <p>1'b0: reset</p>	1'b1

		1'b1: reset release	
[8]	RW	<p>soft_RST_BUS2_N</p> <p>Software reset on-chip bus2 module</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	1'b1
[7]	RW	<p>soft_RST_BUS1_N</p> <p>Software reset on-chip bus1 module</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	1'b1
[6]	RW	<p>soft_RST_APB_N</p> <p>software reset abp bridge module</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	1'b1
[5]	RW	<p>soft_RST_MEM_MNG_N</p> <p>Software reset mem_mng module (internal use, do not modify)</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	1'b1
[4]	RW	<p>soft_RST_DMA_N</p> <p>software reset dma module</p> <p>1'b0: reset</p> <p>1'b1: reset release</p>	1'b1
[3]	RW	soft_RST_SDIO_AHB_N	1'b1

		software reset sdio ahb clock domain module 1'b0: reset 1'b1: reset release	
[2]	RW	soft_RST_SEC_N Software reset security module (internal use, do not modify) 1'b0: reset 1'b1: reset release	1'b1
[1]	RW	soft_RST_MAC_N Software reset mac module (internal use, do not modify) 1'b0: reset 1'b1: reset release	1'b1
[0]	RW	soft_RST_BBP_N Software reset bbp module (internal use, do not modify) 1'b0: reset 1'b1: reset release	1'b1

5.4.5 Clock Divider Configuration Register

Table 11 Clock Divider Configuration Register

bit access		Instructions	reset value
[31]	RW	divide_freq_en When it is necessary to reconfigure cpu_clk_divider, wlan_clk_divider, bus2_syncdn_factor, When sdadc_idiv, set this register, the hardware will automatically update the above four parameters to the frequency divider, and then clear this register	1'b0

		<p>register.</p> <p>1'b0: The frequency division factor has taken effect</p> <p>1'b1: Require hardware to update frequency division parameters</p> <p>Note: When the dividing factor is configured here, when Divide_freq_en is valid, all factors must be valid</p>	
[30:28]		reserve	
[27:24] RW		<p>Peripheral_divider</p> <p>160M clock division factor:</p> <p>Divided by the DPLL as the clock source. The frequency division factor is the assigned frequency division value. The divided output should be 160MHz.</p> <p>The DPLL output is 480MHz and should be configured to 3.</p>	4'h3
[23:16] RW		<p>bus2_syncdn_factor</p> <p>The clock ratio between bus1 and bus2 should be N:1</p> <p>Among them, N is an integer. In the actual adjustment, it mainly depends on the ratio of the operating frequency of the CPU and the clock frequency of bus2.</p> <p>Since the default cpu uses 80MHz clock and bus2 uses 40MHz clock, then N=2</p>	8'h2
[15 : 8] RW		<p>wlan_clk_divider</p> <p>The clock from the PLL is divided and sent to the wlan system. This register is the frequency division factor, which is >=2.</p> <p>The default frequency division factor is 3, that is, the 480MHz output of pll is divided by 3 to obtain a 160MHz clock, which is used as the root</p> <p>The node clock is sent to wlan (wlan continues to divide the frequency to obtain a more detailed low-frequency clock);</p> <p>Note 1: If the WLAN system needs to work normally, this clock needs to be fixed at 160MHz; if the WLAN system is turned off, then this clock can be downclocked to save power. This clock must not be configured higher than 160MHz.</p> <p>Note 2: The secondary bus clock and APB clock are divided by 4 for this clock;</p>	8'h3

[7 : 0] RW		<p>cpu_clk_divider</p> <p>The clock from the PLL is divided and sent to the CPU. This register is the frequency division factor, which is >=2.</p> <p>The default frequency division factor is 6, that is, after the reset is released, the 480MHz clock output by the PLL is divided by 6 and sent to the cpu is the 80MHz clock. When you need to adjust the clock required by the cpu, you can reconfigure this parameter</p>	8'h6
------------	--	---	------

5.4.6 Debug Control Register

Table 12 Clock Select Register

bit access		Instructions	reset value
[16]	RW	<p>JTAG enable</p> <p>1'b0: Disable JTAG debugging</p> <p>1'b1: Enable JTAG debug function</p>	1'b0
[15:8] RW		<p>sd_adc_div</p> <p>sigma-delta ADC clock division factor:</p> <p>Divide by 40MHz as the clock source. The frequency division factor is the assigned frequency division value.</p> <p>After configuring this register, Divide_freq_en in the register clk_divider must be configured to take effect;</p>	8'd10
[7]	RW	RSV	1'b0
[6]	RW	<p>qflash_clk_sel</p> <p>QSPI_FLASH clock selection</p> <p>1: Use 80MHz;</p>	1'b0

		0: use 40MHz;	
[5]	RW	<p>gpsec_sel</p> <p>GPSEC clock selection</p> <p>1: Use 160MHz;</p> <p>0: use 80MHz;</p>	1'b0
[4]	RW	<p>rsa_sel</p> <p>RSA clock selection</p> <p>1: Use 160MHz;</p> <p>0: use 80MHz;</p>	1'b0
[3:0] RW		reserved, do not modify	4'd0

5.4.7 I2S Clock Control Register

Table 13 I2S Clock Control Register

bit access		Instructions	reset value
[31:18]		reserve	
[17:8] RW		<p>BCLKDIV</p> <p>BCLK splitter: $F_{BCLK} = F_{I2SCLK} / BCLKDIV$</p> <p>Note: If EXTAL_EN is not selected and internal PLL is used then $F_{I2SCLK} = F_{CPU}$ (same as CPU frequency) same).</p> <p>Assuming $F_{CPU} = 160MHz$, $F_{I2SCLK} = \text{external crystal frequency}$ when WXTAL_EN is enabled,</p> <p>$BCLKDIV = \text{round}(F_{I2SCLK}/(Fs*W*F))$</p> <p>Where Fs is the sampling frequency of the audio data, W is the word width;</p>	10'b0

		<p>F = 1 when the data is mono;</p> <p>F = 2 when the data is stereo.</p> <p>For example, if the internal PLL is used and the data width is 24 bits, the format is stereo and the sampling frequency is 128KHz, BCLKDIV should be configured as $(160 * 10e6 / 128) * 10e3 * \frac{1}{\text{twenty four}} * 2 = 10'h1a$.</p>	
[7 : 2] RW		<p>MCLKDIV</p> <p>If an external clock is selected, this MCLK divider is used to generate the appropriate MCLK frequency.</p> <p>$F_{mclk} = F_{I2SCLK} / (2 * MCLKDIV)$;</p> <p>$F_{I2SCLK}$ is the external clock when $MCLKDIV = 0$;</p> <p>$F_{mclk} = F_{I2SCLK}$ when $MCLKDIV >= 1$;</p> <p>Note: F_{mclk} should be configured as $256 * fs$, where fs is the sampling frequency.</p>	6'b0
[1]	RW	<p>MCLKEN</p> <p>MCLK enable switch</p> <p>1'b0: MCLK disabled</p> <p>1'b1: enable MCLK</p>	1'b0
[0]	RW	<p>EXTAL_EN</p> <p>External clock selection, choose whether to use the internal I2S block clock or external clock</p> <p>1'b0: Internal clock</p> <p>1'b1: External clock</p> <p>Note: When using an external clock, the external clock must be $2^N * 256 fs$, where fs is the sampling frequency, N must be Must be an integer.</p>	1'b0

5.4.8 Reset Status Register

Table 14 Reset Status Register

bit access		Instructions	reset value
[31:18]		reserve	
[17:8]	WO	CPU soft reset state clear Write 1 to clear CPU soft Reset Status.	1'b0
[7 : 2] WO		Wdog soft reset state clear Writing a 1 clears the Wdog Reset Status.	1'b0
[1]	RO	CPU soft reset state 1: The CPU has generated a soft reset; 0: The CPU does not generate a soft reset;	1'b0
[0]	RO	Wdog reset state 1: Wdog generated Reset; 0: Wdpg does not generate Reset	1'b0

6 DMA modules

6.1 Function overview

DMA is used to provide high-speed data transfer between peripherals and memory and between memory and memory. Can operate without any CPU

Fast data movement via DMA without The CPU resources saved in this way do not affect the operations of other instructions performed by the CPU.

DMA is mounted on the AHB bus, supports up to 8 channels, 16 hardware peripheral request sources, and supports linked list structure and register control.

6.2 Main Features

- ÿ Amba2.0 standard bus interface, 8 DMA channels
- ÿ Support DMA operation based on memory linked list structure
- ÿ Support 16 hardware peripheral request sources
- ÿ Support 1, 4-burst operation mode
- ÿ Support byte, half-word, word as unit transfer operation
- ÿ Support source and destination address unchanged or sequentially incremented or configurable to cycle operations within a predefined address range
- ÿ Supports data transfer methods from memory to memory, memory to peripherals, and peripherals to memory

6.3 Functional Description

6.3.1 DMA channel

W800 supports a total of 8 DMA channels, DMA channels do not interfere with each other and can run at the same time. Request different data streams can choose different DMA channel.

Each DMA channel is allocated in a different register address offset segment, you can directly select the address segment of the corresponding channel for configuration. Do not

The register configuration method of the same channel is exactly the same.

Table 15 DMA address assignment

DMA base address	0x4000 0800
DMA_CH0	Offset (0x10~0x38)
DMA_CH1	Offset (0x40~0x68)
DMA_CH2	Offset (0x70~0x98)
...	...
DMA_CH7	Offset (0x160~0x188)

6.3.2 DMA data flow

Eight DMA channels enable unidirectional data transfer link between source and destination.

The source and destination addresses of the DMA can be set to remain unchanged, incremented or cyclic after each DMA operation is completed:

ÿ DMA_CTRL[2:1] controls how the source address changes after each DMA operation;

ÿ DMA_CTRL[4:3] controls how the destination address changes after each DMA operation.

DMA can set the handling unit of byte, half-word and word, and the final quantity of data to be handled is an integer multiple of the handling unit.

DMA_CTRL[6:5] to set.

DMA can set how many units of data to transfer each time through burst, and choose to transfer 1 or 4 units at a time through DMA_CTRL[7].

Bit data, if DMA_CTRL[6:5] is set to word and burst is set to 4, then 4 words of data are transferred each time.

DMA can set the number of Bytes to start DMA transfer each time, the maximum is 65535 Bytes, which can be set by DMA_CTRL[23:8].

6.3.3 DMA Cycling Mode

The DMA loop address mode means that after the source and destination addresses of DMA are set, after the data transfer reaches the set loop boundary, it will jump to

The loop start address, and this loop executes until the set transfer byte is reached.

The source and destination addresses of the circular address mode need to be set with the SRC_WRAP_ADDR and DEST_WRAP_ADDR registers, and are set by the SRC_WRAP_ADDR and DEST_WRAP_ADDR registers.

WRAP_SIZE to set the loop length value.

6.3.4 DMA transfer mode

DMA supports 3 transfer modes:

ÿ RAM to RAM

Both the source address and the destination address are configured as memory addresses that need to be transferred, and DMA_MODE[0] is set to 0, in software mode.

ÿ Memory to Peripherals

The source address is set to the memory address, the destination address is set to the peripheral address, DMA_MODE[0] is set to 1, the hardware mode,

DMA_MODE[5:2] selects the peripheral used.

ÿ Peripherals to memory

The source address is set to the peripheral address, the destination address is set to the memory address, DMA_MODE[0] is set to 1, the hardware mode,

DMA_MODE[5:2] selects the peripheral used.

6.3.5 DMA peripheral selection

When using the transfer method of peripheral to memory or memory to peripheral, in addition to the corresponding peripheral needs to be set to DMA TX or RX,

DMA_MODE[5:2] also needs to select the corresponding peripheral.

Note: Because there are 3 UART ports in total, when UART uses DMA, it is necessary to select the corresponding port through UART_CH[1:0].

UART.

6.3.6 DMA linked list mode

DMA supports linked list working mode. Through the linked list mode, when DMA transfers the current linked list memory data, we can advance to the next

The data is filled in each linked list. After the DMA finishes moving the current linked list, it judges that the next linked list is valid, and can directly move the data of the next linked list.

The linked list method can effectively improve the efficiency of DMA and CPU cooperation.

Linked list operation mode: Set the DMA to linked list working mode through the DMA_MODE[1] register, and then set the DESC_ADDR register

is the starting address of the linked list structure, and then enables DMA through the CHNL_CTRL register. When the DMA process finishes moving the current memory

After that, the software informs the DMA that there is still valid data in the linked list by setting the valid flag, and the DMA processes the data according to the valid flag of the linked list.

A data to be moved.

6.3.7 DMA Interrupts

An interrupt can be generated when the DMA transfer is completed or burst, and the INT_MASK register can mask the interrupt corresponding to the DMA channel.

When the corresponding DMA interrupt is generated, the current interrupt status can be queried through the INT_SRC register to indicate what is currently generating the interrupt.

The corresponding status bits need to be cleared by software by writing 1 to 1.

6.4 Register Description

6.4.1 Register List

Table 16 DMA register list

offset address	name	abbreviation	access	describe	reset value
0X0000	Interrupt Mask Register	INT_MASK	RW	sets the DMA interrupt that needs to be masked	0X0000_FFFF
0X0004	Interrupt Status Register	INT_SRC	RW	indicates the current DMA interrupt status	0X0000_0000
0X0008	DMA channel selection register DMA_CH		RW	UART peripheral to select which UART	0X0000_0000

0X000C Reserved					
DMA CHNL0 registers					
0X0010	DMA Source Address Register	SRC_ADDR	RW DMA	transfer source address	0X0000_0000
0X0014	DMA Destination Address Register	DEST_ADDR	RW DMA	transfer destination address	0X0000_0000
0X0018	DMA loop source start address register	SRC_WRAP_ADDR	RW DMA	transfer source address in loop mode	0X0000_0000
0X001C	DMA loop destination start address register device	DEST_WRAP_ADD	DMA	transfer destination in RW circular mode	0X0000_0000
0X0020	DMA Cycle Length Register	WRAP_SIZE	DMA	cycl boundary in RW cycle mode	0X0000_0000
0X0024	DMA Channel Control Register	CHNL_CTRL	RW	Current channel DMA start and stop	0X0000_0000
0X0028	DMA Mode Select Register	DMA_MODE	RW	sets how DMA works	0X0000_0000
0X002C	DMA Data Flow Control Register	DMA_CTRL	RW	set DMA transfer data stream	0X0000_0000
0X0030	DMA transfer bytes register	DMA_STATUS	RO	Get the current number of bytes transferred	0X0000_0000
0X0034	DMA linked list entry address register	DESC_ADDR	RW DMA	linked list address entry address setting	0X0000_0000
0X0038	DMA current destination address register	CUR_DEST_ADDR	RO	The address of the current DMA operation	0X0000_0000
DMA CHNL1 registers					
0X0040 - 0X0068	Same as DMA CHNL0 registers				
DMA CHNL2 registers					
0X0070 - 0X0098	Same as DMA CHNL0 registers				
DMA CHNL3 registers					
0X00A0 - same as DMA CHNL0 registers					

0X00C8	
DMA CHNL4 registers	
0X00D0 - 0X00F8	Same as DMA CHNL0 registers
DMA CHNL5 registers	
0X0100 - 0X0128	Same as DMA CHNL0 registers
DMA CHNL6 registers	
0X0130 - 0X0158	Same as DMA CHNL0 registers
DMA CHNL7 registers	
0X0160 - 0X0188	Same as DMA CHNL0 registers

6.4.2 Interrupt Mask Register

Table 17 DMA Interrupt Mask Register

bit access		Instructions	reset value
[31:16]		reserve	
[15]	RW	channel7 transfer_done interrupt mask, active high.	1'b1
[14]	RW	channel7 burst_done interrupt mask, active high.	1'b1
[13]	RW	channel6 transfer_done interrupt mask, active high.	1'b1
[12]	RW	channel6 burst_done interrupt mask, active high.	1'b1

[11]	RW	channel5 transfer_done interrupt mask, active high.	1'b1
[10]	RW	channel5 burst_done interrupt mask, active high.	1'b1
[9]	RW	channel4 transfer_done interrupt mask, active high.	1'b1
[8]	RW	channel4 burst_done interrupt mask, active high.	1'b1
[7]	RW	channel3 transfer_done interrupt mask, active high.	1'b1
[6]	RW	channel3 burst_done interrupt mask, active high.	1'b1
[5]	RW	channel2 transfer_done interrupt mask, active high.	1'b1
[4]	RW	channel2 burst_done interrupt mask, active high.	1'b1
[3]	RW	channel1 transfer_done interrupt mask, active high.	1'b1
[2]	RW	channel1 burst_done interrupt mask, active high.	1'b1
[1]	RW	channel0 transfer_done interrupt mask, active high.	1'b1
[0]	RW	channel0 burst_done interrupt mask, active high.	1'b1

6.4.3 Interrupt Status Register

Table 18 DMA Interrupt Status Register

bit access		Instructions	reset value
[31:16]		reserve	
[15]	RW	channel7 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	1'b0
[14]	RW	channel7 burst_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA burst is complete.	1'b0
[13]	RW	channel6 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	1'b0
[12]	RW	channel6 burst_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA burst is complete.	1'b0
[11]	RW	channel5 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	1'b0

[10]	RW	channel5 burst_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA burst is complete.	1'b0
[9]	RW	channel4 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	1'b0
[8]	RW	channel4 burst_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA burst is complete.	1'b0
[7]	RW	channel3 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	1'b0
[6]	RW	channel3 burst_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA burst is complete.	1'b0
[5]	RW	channel2 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	1'b0
[4]	RW	channel2 burst_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA burst is complete.	1'b0
[3]	RW	channel1 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	1'b0
[2]	RW	channel1 burst_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA burst is complete.	1'b0
[1]	RW	channel0 transfer_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA transfer is complete.	1'b0
[0]	RW	channel0 burst_done interrupt status, write 1 to clear 0. An interrupt is generated when the DMA burst is complete.	1'b0

6.4.4 UART selection register

Table 19 UART select register

bit access		Instructions	reset value
[31:24]		reserve	
[23:8] RW		<p>dma req clear:</p> <p>Write 1 to each bit to clear the corresponding dma req request. Self-cleaning.</p> <p>For example, writing 1 to bit 23 will clear the 15th corresponding dma request in dma_sel;</p> <p>Writing 1 to bit 8 will clear the 0th dma request in dma_sel - uart_rx_req;</p>	16'd0
[2 : 0] RW		<p>Uart dma channel selection:</p> <p>3'd0: uart0 module dma channel access dma</p>	3'h0

		3'd1: uart1 module dma channel access dma 3'd2: uart2/7816 module dma channel access dma 3'd3: uart3 module dma channel access dma 3'd4: uart4 module dma channel access dma 3'd5: uart5 module dma channel access dma	
--	--	--	--

6.4.5 DMA Source Address Register

Table 20 DMA Source Address Register

bit access		Instructions	reset value
[31: 0] RW		In acyclic mode, the source address, peripheral address or memory address of the DMA transfer	32'h0

6.4.6 DMA Destination Address Register

Table 21 DMA Destination Address Register

bit access		Instructions	reset value
[31: 0] RW		In acyclic mode, the destination address, peripheral address or memory address of DMA transfer	32'h0

6.4.7 DMA loop source start address register

Table 22 DMA loop source start address register

bit access		Instructions	reset value
[31: 0] RW		In circular mode, the starting address of the source address, peripheral address or memory address of the DMA transfer	32'h0

6.4.8 DMA loop destination start address register

Table 23 DMA loop destination start address register

bit access		Instructions	reset value
[31: 0] RW		In circular mode, the starting address, peripheral address or memory address of the destination address of DMA transfer	32'h0

6.4.9 DMA Cycle Length Register

Table 24 DMA Cycle Length Register

bit access		Instructions	reset value
[31:16] RW		In loop mode, the loop length of the DMA destination address. DMA moves data sequentially from the start address. When the number of bytes of data to be moved reaches this set value, it will jump Go to the cycle start address and continue to move data from the start address	16'h0
[15:0] RW		In loop mode, the DMA source address loop length.	16'h0

6.4.10 DMA Channel Control Register

Table 25 DMA Channel Control Register

bit access		Instructions	reset value
[31:2]		reserve	
[1]	RW	dma_stop Stop dma operation, active high. The DMA stops after completing the current burst operation and clears chnl_on at the same time. software should be based on chnl_on is 0 to determine that the dma has completely stopped.	1'b0
[0]	RW	chnl_on	1'b0

		<p>Start the current channel DMA conversion, active high.</p> <p>It is automatically cleared to 0 after Dma is completed and the conversion or setup stops.</p>	
--	--	---	--

6.4.11 DMA Mode Select Register

Table 26 DMA Mode Select Register

bit access		Instructions	reset value
[31:7]		reserve	
[6]	RW	<p>chain_link_en</p> <p>Valid in linked list mode, indicating whether dma continues to read and process subsequent chains after processing the first linked list surface.</p> <p>If it is 1, update the next_desc_addr in the linked list and continue reading the next linked list until the linked list where vld is 0; if it is 0, the processing will stop after completing the current linked list.</p>	1'b0
[5 : 2]	RW	<p>dma_sel</p> <p>Choice of 16 dma_reqs.</p> <p>4'd0: uart rx dma req</p> <p>4'd1: uart tx dma req</p> <p>4'd2: pwm_cap0_req</p> <p>4'd3: pwm_cap1_req</p> <p>4'd4: LS_SPI rx dma req</p> <p>4'd5: LS_SPI tx dma req</p> <p>4'd6: SD_ADC chnl0 req</p> <p>4'd7: SD_ADC chnl1 req</p>	4'h0

		4'd8: SD_ADC chnl2 req 4'd9: SD_ADC chnl3 req 4'd10: I2S RX req 4'd11: I2S TX req 4'd12: SDIO_HOST req	
[1]	RW	chain_mode 1'b0: use normal mode 1'b1: use linked list mode	1'b0
[0]	RW	dma_mode 1'b0: Software mode. 1'b1: Hardware mode.	1'b0

6.4.12 DMA Data Flow Control Register

Table 27 DMA Data Flow Control Register

bit access		Instructions	reset value
[31:24]		reserve	
[23:8] RW		total_byte The total number of bytes to be operated on. It needs to be consistent with the data_size configuration, that is, if it is a word operation, then Should be configured as an integer multiple of 4; if it is a halfword operation, it should be configured as an integer multiple of 2.	16'h0
[7]	RW	burst_size Set how many units of data the DMA transfers at a time 1'b0: burst is 1	1'b0

		1'b1: burst is 4 When the last burst size exceeds the number of remaining transfers, use the burst size as the size of the remaining data Small.	
[6 : 5] RW		data_size Set the handling unit for DMA 2'h0: byte 2'h1: half_word 2'h2: word 2'h3: reserved	2'h0
[4 : 3] RW		dest_addr_inc 2h0: The destination address remains unchanged after each operation; 2h1: The destination address is automatically accumulated after each operation. 2'h2: reserved 2'h3: Loop operation, the destination address is automatically accumulated after each operation, and it jumps to the start of the loop when it reaches the defined loop boundary. starting address.	2'h0
[2 : 1] RW		src_addr_inc 2h0: The source address remains unchanged after each operation; 2h1: The source address is automatically accumulated after each operation. 2'h2: reserved 2'h3: Loop operation, the source address is automatically accumulated after each operation, and jumps to the start of the loop when it reaches the defined loop boundary address.	2'h0
[0]	RW	auto_reload	1'b0

		When the current DMA transfer is completed, the next DMA transfer will be performed automatically according to the current DMA configuration.	
--	--	---	--

6.4.13 DMA transfer bytes register

Table 28 DMA Transfer Bytes Register

bit access		Instructions	reset value
[31:16]		reserve	
[15:0] RW		<p>transfer_cnt</p> <p>The number of bytes currently transferred.</p> <p>Each time the dma is restarted (chnl_on is set to 1), it is cleared to 0, and the counting is restarted.</p>	16'h0

6.4.14 DMA linked list entry address register

Table 29 DMA linked list entry address register

bit access		Instructions	reset value
[31: 0] RW		<p>desc_addr</p> <p>When the linked list is enabled, it is used as the entry address of the linked list. After each transfer of the linked list is completed, the base of the next linked list is updated to this register.</p>	32'h0

6.4.15 DMA current destination address register

Table 30 DMA current destination address register

bit access		Instructions	reset value
[31:0] RO		<p>current_dest_addr</p> <p>The destination address of the current DMA operation.</p>	32'h0

		当软件停止dma时，可以通过查看此寄存器获悉dma将要操作的目的地址。	
--	--	-------------------------------------	--

7 Universal hardware encryption module

7.1 Function overview

The encryption module automatically completes the encryption of the source address space data of the specified length, and automatically writes the encrypted data back to the specified destination address after completion.

time; supports SHA1/MD5/RC4/DES/3DES/AES/CRC/TRNG.

7.2 Main Features

- ÿ Support SHA1/MD5/RC4/DES/3DES/AES/CRC/TRNG encryption algorithm
- ÿ DES/3DES supports ECB and CBC modes
- ÿ AES supports ECB, CBC and CTR modes
- ÿ CRC supports four modes: CRC8, CRC16_MODBUS, CRC16_CCITT and CRC32
- ÿ CRC supports input/output reverse
- ÿ SHA1/MD5/CRC supports continuous multi-packet encryption
- ÿ Built-in true random number generator, also supports seed to generate pseudo-random numbers

7.3 Functional Description

7.3.1 SHA1 encryption

Hardware SHA1 calculation can be performed on consecutive multiple packets of data, the calculation result is stored in the register, and the encryption result of the previous packet can be used as the encryption result of the next packet.

initial value.

7.3.2 MD5 encryption

Hardware MD5 calculation can be performed on consecutive multiple packets of data, the calculation result is stored in the register, and the encryption result of the previous packet can be used as the initial value of the next packet.

initial value.

7.3.3 RC4 encryption

Supports RC4 encryption and decryption.

7.3.4 DES encryption

Support DES encryption and decryption, support ECB and CBC two modes.

7.3.5 3DES encryption

Support 3DES encryption and decryption, support ECB and CBC two modes.

7.3.6 AES encryption

Support AES encryption and decryption, support ECB, CBC and CTR three modes.

7.3.7 CRC encryption

The hardware CRC calculation can be performed on consecutive multi-packet data, the calculation result is stored in the register, and the encryption result of the previous packet can be used as the initial value of the next packet.

It supports four modes: CRC8, CRC16_MODBUS, CRC16_CCITT and CRC32, and supports input/output inversion.

The calculation formula of CRC32 is as follows:

1. CRC-32: 0x04C11DB7

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

Commonly used in ZIP, RAR, IEEE 802 LAN/FDDI, IEEE 1394, PPP-FCS and other protocols.

2. CRC-16: supports two polynomials

2.1: 0X1021

$$X^{16} + X^{12} + X^5 + 1$$

Commonly used in ISO HDLC, ITU X.25, V.34/V.41/V.42, PPP-FCS CCITT and other protocols.

2.2: 0X8005

X16 + X15 + X2 + 1

Commonly used in USB, ANSI X3.28, SIA DC-07 and other protocols.

3. CRC-8: 0X207

x8+x2+x1+1

7.3.8 TRNG Random Number Generator

A true random number generator module is integrated into the W800 system. Divided into analog module and digital post-processing module. The analog module outputs a random clock ad_trng_clks and random number ad_trng_dout, the digital post-processing module is used to eliminate the bias and autocorrelation of random numbers. Related control

The control register is in the GPSEC register list.

The basic operation process is as follows:

1. Enable TRNG_EN and set TRNG_SEL to 1, so that GPSEC register 0x48 displays the output value of TRNG mode at this time

The pseudo module starts to output random clocks and random signals. The signal sampled by the first 8 clocks is used as the initial state of the LFSR to initialize the LFSR

chain, the data sampled by each random clock is post-processed by the XOR CHAIN and LFSR registers, and then shifted and stored in the register.

in the server TRNG_RANDOM.

2. Software can read random value through GPSEC register 0x48. Digital postprocessing when register TRNG_DIG_BYPASS is set to 1

The module stops working and directly stores the output value of the analog module into the result register TRNG_RANDOM.

7.4 Register Description

7.4.1 Register List

Table 31 Encryption module register list

offset address	name	abbreviation	access	describe	reset value
0X0000	Source address register	SRC_ADDR	RW RC4	SHA1/AES/DES/3DES/CRC/MD 5 Multiplexing source address	0X0000_0000
0X0004	Destination address register	DEST_ADDR	RW RC4	AES/DES/3DES multiplexing destination address 0X0000_0000	0X0000_0000
0X0008	Configuration register	GPSEC_CFG	RW Generic	Hardware Cryptographic Module Configuration Register	0X0000_0000
0X000C	Control Register	GPSEC_CTRL	RW Generic	Hardware Cryptographic Module Control Register	0X0000_0000
0X0010	Key 0 low register KEY00		RW Key0	Low 32-bit first input key (RC4/AES/DES/3DES), multiplexed CRC Ci	0X0000_0000
0X0014	Key 0 high register KEY01		RW Key0	High 32-bit first input key (RC4/AES/DES/3DES)	0X0000_0000
0X0018	Key 1 low register KEY10		RW Key1	lower 32-bit second input key (RC4/AES//3DES)	0X0000_0000
0X001C	Key 1 high register KEY11		RW Key1	High 32-bit second input key (RC4/AES//3DES)	0X0000_0000
0X0020	Key 2 low register	KEY20	RW Key2	The third input key of the lower 32 bits (3DES), multiplex iv1 low 32-bit input initial Vector (AES)	0X0000_0000

0X0024 Key 2 high register	KEY21	RW Key2	High 32-bit third input key (3DES), multiplex iv1 high 32-bit input initial Vector (AES)	0X0000_0000
0X0028	Initial vector 0 low register <small>device</small>	IV00	RW IV0 low 32-bit input initial vector (AES/DES/3DES)	0X0000_0000
0X002C	Initial vector 0 high order register <small>device</small>	IV01	RW IV0 upper 32-bit input initial vector (AES/DES/3DES)	0X0000_0000
0X0030 Status Register	GPSEC_STS	RW Generic	Hardware Cryptographic Module Status Register	0X0000_0000
0X0034 Summary 0 register	SHA1-DIGEST0	RW sha1	digest0/MD5-digest0	0X6745_2301
0X0038 Summary 1 Register	SHA1-DIGEST1	RW sha1	digest1/MD5-digest1	0XEFC0_AB89
0X003C Summary 2 Register	SHA1-DIGEST2	RW sha1	digest2/MD5-digest2	0X98BA_DCFE
0X0040 Summary 3 register	SHA1-DIGEST3	RW sha1	digest3/MD5-digest3	0X1032_5476
0X0044 Summary 4 Register	SHA1-DIGEST4	RW sha1	digest4/CRC	0XC3D2_E1F0
0X0048 RNG result	RNG_RESULT	RW RNG	output	0X0000_0000
0X004C	Key 3 low register Key30	RW Key3	lower 32-bit third input key (RC4's 256bit mode)	0X0000_0000
0X0050	Key 3 low register Key31	RW Key3	high 32-bit third input key (RC4's 256bit mode)	0X0000_0000
0X0054 TRNG configuration	TRNG_CR	RW True Random Number Generator Configuration Selection		0X40

7.4.2 Configuration Registers

Table 32 Cryptographic Module Configuration Registers

bit access		Instructions	reset value
[31]	RW	<p>RC4_128_256</p> <p>0: indicates that RC4 encryption/decryption is performed according to the 128bit block length;</p> <p>1: Flag RC4 encryption/decryption is performed according to 256bit block length.</p>	1'b0
[30]	RW	<p>RNG start</p> <p>1'b0: do not start RNG</p> <p>1'b1: start RNG</p>	1'b0
[29]	RW	<p>RNG Load_seed</p> <p>Hardware automatically reset to 0</p> <p>1'b0: The random number generator will be seeded with zero by default to generate a random number of the corresponding number of digits</p> <p>1'b1: Start generating random numbers after the seed is loaded</p>	1'b0
[28]	RW	<p>RNG switch</p> <p>controls the number of bits to generate random numbers,</p> <p>1'b0: 16 bits</p> <p>1'b1: 32 bits</p>	1'b0
[27]	RW	<p>des_soft_reset</p> <p>des is automatically cleared to 0 by hardware after the soft reset is completed</p> <p>1'b0: No soft reset is generated and the current state is not changed</p> <p>1'b1: The encryption algorithm is reset to the initial state by software</p>	1'b0
[26]	RW	<p>aes_soft_reset</p> <p>aes is automatically cleared to 0 by hardware after the soft reset is completed</p> <p>1'b0: No soft reset is generated and the current state is not changed</p>	1'b0

		1'b1: The encryption algorithm is reset to the initial state by software	
[25]	RW	<p>rc4_soft_reset</p> <p>rc4 After the soft reset is completed, the hardware is automatically cleared to 0</p> <p>1'b0: No soft reset is generated and the current state is not changed</p> <p>1'b1: The encryption algorithm is reset to the initial state by software</p>	1'b0
[twenty four]	RW	<p>crc_datarev</p> <p>1'b0: CRC input data is not reversed</p> <p>1'b1: CRC input data reverse</p>	1'b0
[twenty three]	RW	<p>crc_chksrev</p> <p>1'b0: CRC output result is not reversed</p> <p>1'b1: CRC output result is reversed</p>	1'b0
[22:21] RW		<p>sub_mode</p> <p>Algorithm type submode selection:</p> <p>0: ECB mode of DES/AES encryption algorithm, CRC8 mode of CRC algorithm can be reused</p> <p>1: CBC of DES/AES cipher algorithm, CRC16_0 mode of reusable CRC algorithm</p> <p>2: CTR mode of AES encryption algorithm, CRC16_1 mode of CRC algorithm can be reused</p> <p>3: MAC mode of AES encryption algorithm, CRC32 of CRC algorithm can be reused</p>	2'b0
[20]	RW	<p>encrypt_decrypt</p> <p>Encryption or decryption mode selection for RC4/AES/DES/3DES algorithm:</p> <p>1'b0: encryption</p> <p>1'b1: decrypt</p>	1'b0
[19]	RW	gpsec_int_mask	1'b0

		1'b0: Do not mask encryption/decryption completion interrupt 1'b1: Shield encryption/decryption completion interrupt	
[18:16] RW		<p>cypher_mode</p> <p>Cryptographic Algorithm Type</p> <p>3'b000: RSV</p> <p>3'b001: RC4</p> <p>3'b010: SHA1</p> <p>3'b011: AES</p> <p>3'b100: DES</p> <p>3'b101: 3DES</p> <p>3'b110: CRC</p> <p>3'b111: MD5</p>	3'b0
[15:0] RW		<p>total_byte</p> <p>The total number of bytes required for encryption and decryption operations.</p>	16'h0

7.4.3 TRNG Control Register

Table 33 TRNG Module Control Register

bit access		Instructions	reset value
[31:7]		reserve	
[6]	RW	<p>TRNG_INT_MASK</p> <p>TRNG interrupt mask</p>	1'b1

		0: TRNG module reports interrupt; 1: The TRNG module does not report interrupts;	
[5:3] RW		TRNG_CP TRNG module control signal	3'd0
[2]	RW	TRNG_DIG_BYPASS TRNG digital post-processing module bypass: 0: TRNG module for post-processing; 1: The TRNG module does not perform post-processing;	1'b0
[1]	RW	TRNG_SEL: RNG output selection signal. 0: Register 0x48 displays the output result of the pseudo-random module; 1: Register 0x48 displays the TRNG output result;	1'b0
[0]	RW	TRNG_EN: TRNG module enable signal. Highly effective. 0: TRNG module stops; 1: The TRNG module starts to work;	1'b0

7.4.4 Control Register

Table 34 Cryptographic Module Control Registers

bit access		Instructions	reset value
[31:2]		reserve	

[1]	RW	sec_stop Stop currently ongoing encryption and decryption operations 1'b0: invalid 1'b1: Encryption/decryption stop	1'b0
[0]	RW	sec strt Start encryption and decryption, after completing the number of bytes of encryption and decryption operations, the hardware will automatically clear 0 1'b0: Do not start encryption/decryption 1'b1: start encryption/decryption	1'b0

7.4.5 Status Register

Table 35 Cryptographic Module Status Register

bit access		Instructions	reset value
[31:17]		reserve	
[16]	RW	int_flag Software write 1 to clear 1'b0: do not generate encryption/decryption completion interrupt 1'b1: Generate encryption/decryption completion interrupt	1'b0
[15:0] RO		transfer_cnt The number of bytes currently encrypted. It is cleared to 0 each time the encryption and decryption is restarted, and the counting starts again.	16'h0

8 RSA encryption module

8.1 Function overview

RSA operation hardware coprocessor, providing Montgomery (F1OS algorithm) modular multiplication function. Implementing RSA Algorithm with RSA Software Library

Law. 128-bit to 2048-bit modulo multiplication is supported.

8.2 Main Features

- ÿ Support 128-bit to 2048-bit modulo multiplication (the modulo multiplication length is an integer multiple of 32 bits)

- ÿ Support D*D; X*Y; D*Y; X*X and other 4 modulo multiplication modes

8.3 Functional Description

8.3.1 Modular multiplication function

RSA operation hardware coprocessor, providing Montgomery (F1OS algorithm) modular multiplication function. Implement RSA together with RSA software library

algorithm. 128-bit to 2048-bit modulo multiplication is supported.

8.4 Register Description

8.4.1 Register List

Table 36 RSA register list

offset address name		Abbreviated access		describe	reset value
0X0000~0X00FC	Data X register	XBUF	RW Data	X Register	
0X0100~0X01FC	Data Y register	YBUF	RW Data	Y Register	
0X0200~0X02FC	Data M register	MBUF	RW Data	M Register	
0X0300~0X03FC	Data D register	DBUF	RW Data	D Register	

0X0400	RSA Control Register RSA	ACON RW		RSA Control Register	0X0000_0000
0X0404	Parameter MC register	RSAMC WO parameter	MC register		0X0000_0000
0X0408	Parameter N register	RSAN	RW parameter N register		0X0000_0000

8.4.2 Data X Register

XBUF corresponds to the buffer of data X (2048bit), and the corresponding haddr value is 0000h~00fch. The corresponding rules are as follows:

Table 37 RSA Data X Register

000h	004h	008h	00f8h	00fch
X[31:0]	X[63:32]	X[95:64]	X[2015:1984]	X[2047:2016]

8.4.3 Data Y register

YBUF corresponds to the buffer of data Y (2048bit), and the corresponding haddr value is 0100h~01fch. The corresponding rules are as follows:

Table 38 RSA Data Y Register

0100h	0104h	0108h	01f8h	01fch
Y[31:0]	Y[63:32]	Y[95:64]	Y[2015:1984]	Y[2047:2016]

8.4.4 Data M register

MBUF corresponds to the buffer of data M (2048bit), and the corresponding haddr value is 0200h~02fch. The corresponding rules are as follows:

Table 39 RSA Data M Register

0200h	0204h	0208h	02f8h	02fch
M[31:0]	M[63:32]	M[95:64]	M[2015:1984]	M[2047:2016]

8.4.5 Data D Register

DBUF corresponds to the buffer of data D (2048bit), and the corresponding haddr value is 0300h~03fch. The corresponding rules are as follows:

Table 40 RSA Data D Register

0300h	0304h	0308h	03f8h	03fch
D[31:0]	D[63:32] D[95:64] ...			D[2015:198] 4]	D[2047:2016]]

8.4.6 RSA Control Register

RSACON, RSA control register, the actual physical space is a 32bit register.

Table 41 RSA Control Register

bit access		Instructions	reset value
[31:6]		reserve	
[5]	RW	Modulo multiplication start control bit. The software writes "1" to start the modulo multiplication operation. After the operation is completed, the hardware automatically clears it to "0". 1'b0	
[4]	RW	<p>Provides soft reset function, active high. The software writes "1" to perform a soft reset. After the reset is completed, the hardware automatically clears "0".</p> <p>1. Set parameters MC and N to 0.</p> <p>2. After the modulo multiplication is started (bit5 is set to 1), this bit will be set to 1, and the current operation will be terminated (when bit0 changes to 1).</p> <p>High, indicating that the soft reset command is executed and the operation is terminated), but the internal data buffer (X, Y, M, D) Part of the operation results that have been completed in the .</p>	1'b0
[2 : 3] RW		Modulo multiplication mode selection. 2'b00: X = D*D mod M 2'b01: D = X*Y mod M 2'b10: X = D*Y mod M 2'b11: D = X*X mod M	2'b0
[1]	RW	reserve	1'b0

[0]	RW	Modulo multiplication completion flag, high effective. Set to "1" by hardware and clear to "0" by software. Software writing "1" is invalid. 1'b0
-----	----	---

8.4.7 Parameter MC register

Table 42 RSA parameter MC register

bit access		Instructions	reset value
[31:0] WO		RSAMC corresponds to the parameter MC (32bit). The reset value is all 0s. The read value is all 0s.	32'h0

8.4.8 Parameter N register

RSAN corresponds to parameter N (7bit). The N value is the modulo multiplied length divided by 32. That is, if you call the 1024bit modular multiplication operation, you need to set N = 32. When writing to this register, the lower 7 bits are taken as valid data, and when reading, the upper 25 bits are 0. The reset value is all 0s.

Table 43 RSA parameter N register

bit access		Instructions	reset value
[31:7]		reserve	
[6 : 0] RW		RSAN corresponds to parameter N (7bit). The N value is the modulo multiplied length divided by 32.	7'h0

9 GPIO module

9.1 Function overview

The GPIO controller implements the configuration of the GPIO properties by software, enabling users to conveniently operate the GPIO.

Each GPIO can be individually configured by software, set it as input port, output port, set its floating, pull-up, pull-down state,

Set its rising edge, falling edge, double edge, high level, low level interrupt trigger mode.

9.2 Main Features

- ÿ Support GPIO software configuration
- ÿ Support GPIO interrupt configuration
- ÿ Provides up to 48 GPIOs available

9.3 Functional Description

The GPIO provided in W800 is divided into two groups, one is GPIOA, the other is GPIOB. The starting addresses of GPIOA and GPIOB registers are different.

Same, but function the same.

When the user wants to use a specific IO as a software-controlled GPIO, set the corresponding position in the GPIO multiplexing selection register to 0, i.e.

Can.

The GPIO direction control register is used to control the direction of the GPIO, 1 means the corresponding GPIO is used as an output pin, 0 means the corresponding GPIO as an input pin.

The GPIO pull-up and pull-down control registers are used to control the pull-up and pull-down functions of the corresponding IO.

The GPIO pull-up control register is active low, setting it to 0 means to enable the pull-up function of the corresponding IO, and setting it to 1 means to close the pull-up function.

For the attributes of IO, please refer to the IO multiplexing table.

The GPIO pull-down control register is active high. Setting it to 1 means to open the pull-down function of the corresponding IO, and setting it to 0 means to close the pull-down function.

For the attributes of IO, please refer to the IO multiplexing table.

When the GPIO data register is set to the input state, it represents the level of the input IO. When it is set to the output state, 1 or 0 can be written to specify

IO output level. This register is controlled by the GPIO data enable register, only when the GPIO data enable register is set to 1

time, the GPIO data register can be read and written.

The GPIO module provides input signal detection function. High and low level detection and upper and lower edges can be realized by configuring GPIO interrupt related registers

Jump detection. When the input signal corresponding to IO meets the preset conditions, such as high-level trigger or rising edge trigger, etc., it will trigger

GPIO interrupt is reported to MCU for processing. The MCU needs to clear the corresponding interrupt status to avoid false triggering of the interrupt.

9.4 Register Description

9.4.1 Register List

Table 44 GPIOA register list

offset address	name	Abbreviated access		describe	reset value
0X0000	GPIO data register	GPIO_DATA	RW Read and write GPIO current data		0X180B
0X0004	GPIO data enable register	GPIO_DATA_E N	RW Configure the enable bit of GPIO_DATA		0xFFFF
0X0008	GPIO direction control register	GPIO_DIR	RW configure GPIO direction		0X0000
0X000C	GPIO Pull-Up Control Register	GPIO_PULL_E N	RW configure GPIO pull-up		0xFFFF
0X0010	GPIO multiplexing selection register	GPIO_AF_SEL	RW Configure GPIO alternate function enable bit		0xFFFF

0X0014	GPIO multiplexing selection register 1	GPIO_SF_S1	RW GPIO	alternate function selection bit high address bit	0X0000
0X0018	GPIO multiplexing select register 0	GPIO_AF_S0	RW GPIO	alternate function selection bit low address bit	0X0000
0X001C	GPIO pull-down control register	GPIO_DN_ENA	RW Configure	GPIO pull-down	0X0000
0X0020	GPIO interrupt trigger mode configuration register	GPIO_IS	RW Configure	the interrupt triggering method of GPIO	0X0000
0X0024	GPIO interrupt edge-triggered mode configuration register register	GPIO_IBE	RW Configure	GPIO interrupt edge trigger mode	0X0000
0X0028	GPIO interrupt upper and lower edge trigger configuration register register	GPIO_IEV	RW Configure	GPIO interrupt upper and lower edge trigger or high and low level trigger	0X0000
0X002C	GPIO interrupt enable configuration register	GPIO_IE	RW configure	GPIO interrupt enable	0X0000
0X0030	GPIO Bare Interrupt Status Register	GPIO_RIS	RO Query	GPIO raw interrupt status (before MASK) 0X0000	
0X0034	GPIO masked interrupt status register	GPIO_MIS	RO Query	the interrupt status after GPIO masking (MASK back)	0X0000
0X0038	GPIO interrupt clear control register	GPIO_IC	WO controls	GPIO interrupt clearing	0X0000

Table 45 GPIOB register list

offset address	name	Abbreviated access		describe	reset value
0X0000	GPIO data register	GPIO_DATA	RW Read	and write GPIO current data	0X0000_7304
0X0004	GPIO data enable register	GPIO_DATA_E	RW Configure	the enable bit of GPIO_DATA 0X7FFF_FFFF	
N					
0X0008	GPIO direction control register	GPIO_DIR	RW config	ure GPIO direction	0X0000_0000
0X000C	GPIO Pull-Up Control Register	GPIO_PULL_E	RW config	ure GPIO pull-up	0xFFFF_FFFF
N					

0X0010	GPIO multiplexing selection register	GPIO_AF_SEL RW	Configure GPIO alternate function enable bit 0xFFFF_FFFF	
0X0014	GPIO multiplexing selection register 1	GPIO_SF_S1	RW GPIO alternate function selection bit high address bit 0X0000_0000	
0X0018	GPIO multiplexing select register 0	GPIO_AF_S0 RW	GPIO alternate function selection bit low address bit 0X0000_0000	
0X001C	GPIO pull-down control register	GPIO_DN_ENA RW	Configure GPIO pull-down	0X0000_0000
0X0020	GPIO interrupt trigger mode configuration register	GPIO_IS	RW Configure the interrupt trigger mode of GPIO 0X0000_0000	
0X0024	GPIO interrupt edge-triggered mode configuration register register	GPIO_IBE	RW Configure GPIO interrupt edge trigger mode 0X0000_0000	
0X0028	GPIO interrupt upper and lower edge trigger configuration register register	GPIO_IEV	RW Configure GPIO interrupt to be edge-triggered or High and low level trigger	0X0000_0000
0X002C	GPIO interrupt enable configuration register	GPIO_IE	RW configure GPIO interrupt enable	0X0000_0000
0X0030	GPIO Bare Interrupt Status Register	GPIO_RIS	RO queries GPIO bare interrupt status (MASK forward)	0X0000_0000
0X0034	GPIO masked interrupt status register	GPIO_MIS	RO Query the interrupt status after GPIO masking (after MASK)	0X0000_0000
0X0038	GPIO interrupt clear control register	GPIO_IC	WO controls GPIO interrupt clearing	0X0000_0000

9.4.2 GPIO data register

Table 46 GPIOA data register

bit access		Instructions	reset value
[15:0]	RW	GPIO current data, each BIT corresponds to the corresponding GPIO line	16'h180b

Table 47 GPIOB data register

bit access		Instructions	reset value
[31:0]	RW	GPIO current data, each BIT corresponds to the corresponding GPIO line	32'h7304

9.4.3 GPIO Data Enable Register

Table 48 GPIOA data enable register

bit access		Instructions	reset value
[15:0]	RW	<p>Corresponding to the BIT enable bit of GPIO_DATA, only when the corresponding BIT is 1, the operation of the corresponding bit of GPIO_DATA</p> <p>It is valid only after the operation, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, GPIO_DATA[x] cannot be read or written</p> <p>[x] = 1, GPIO_DATA[x] can be read and written</p>	16'hffff

Table 49 GPIOB data enable register

bit access		Instructions	reset value
[31:0]	RW	<p>Corresponding to the BIT enable bit of GPIO_DATA, only when the corresponding BIT is 1, the operation of the corresponding bit of GPIO_DATA</p> <p>It is valid only after the operation, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, GPIO_DATA[x] cannot be read or written</p> <p>[x] = 1, GPIO_DATA[x] can be read and written</p>	32'h7fff_ffff

9.4.4 GPIO direction control register

Table 50 GPIOA direction control register

bit access		Instructions	reset value
[15:0]	RW	GPIO direction control, each BIT corresponds to the corresponding GPIO line, 1'bx:	16'h0

		[x] = 0, GPIO[x] is input [x] = 1, GPIO[x] is output	
--	--	---	--

Table 51 GPIOB direction control register

bit access		Instructions	reset value
[31:0]	RW	GPIO direction control, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] is input [x] = 1, GPIO[x] is output	32'h0

9.4.5 GPIO pull-up and pull-down control registers

Table 52 GPIOA pull-up control register

bit access		Instructions	reset value
[15:0]	RW	GPIO pull-up control, each BIT corresponds to the corresponding GPIO line, 1'bx: Note: This register is active low [x] = 0, GPIO[x] has pull-up [x] = 1, no pull-up on GPIO[x]	16'hffff

bit access		Instructions	reset value
[15:0]	RW	GPIO pull-down control, each BIT corresponds to the corresponding GPIO line, 1'bx: Note: This register is active high [x] = 1, GPIO[x] has pull-down [x] = 0, GPIO[x] has no pull-down	16'h0000

Table 53 GPIOB pull-up and pull-down control registers

bit access		Instructions	reset value
[31:0]	RW	<p>GPIO pull-up control, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>Note: This register is active low</p> <p>[x] = 0, GPIO[x] has pull-up</p> <p>[x] = 1, no pull-up on GPIO[x]</p>	32'hffff_ffff

bit access		Instructions	reset value
[31:0]	RW	<p>GPIO pull-down control, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>Note: This register is active high</p> <p>[x] = 1, GPIO[x] has pull-down</p> <p>[x] = 0, GPIO[x] has no pull-down</p>	32'h0000_0000

9.4.6 GPIO multiplexing selection register

Table 54 GPIOA multiplexing selection register

bit access		Instructions	reset value
[15:0]	RW	GPIO multiplexing function enable bit, each BIT corresponds to whether the corresponding GPIO multiplexing function is enabled, 1'bx:	16'hffff

		<p>[x] = 0, GPIO[x] alternate function is disabled</p> <p>[x] = 1, GPIO[x] alternate function is enabled</p> <p>When [x] = 1, the alternate function depends on the corresponding BITs of the two registers GPIO_AF_S1 and GPIO_AF_S0</p> <p>status.</p> <p>S1[x] = 0, S0.[x] = 0, alternate function 1 (opt1)</p> <p>S1[x] = 0, S0.[x] = 1, alternate function 2 (opt2)</p> <p>S1[x] = 1, S0.[x] = 0, alternate function 3 (opt3)</p> <p>S1[x] = 1, S0.[x] = 1, multiplexing function 4 (opt4)</p> <p>When [x] = 0, if GPIO_DIR[x] = 0, and GPIO_PULL_EN[x] = 1, the GPIO multiplexing is opt6 analog IO function</p> <p>For the IO multiplexing function, please refer to the chip pin multiplexing relationship</p>	
--	--	--	--

Table 55 GPIOB multiplexing selection register

bit access		Instructions	reset value
[31:0]	RW	<p>GPIO multiplexing function enable bit, each BIT corresponds to whether the corresponding GPIO multiplexing function is enabled, 1'b:</p> <p>[x] = 0, GPIO[x] alternate function is disabled</p> <p>[x] = 1, GPIO[x] alternate function is enabled</p> <p>When [x] = 1, the alternate function depends on the corresponding BITs of the two registers GPIO_AF_S1 and GPIO_AF_S0</p> <p>status.</p> <p>S1[x] = 0, S0.[x] = 0, alternate function 1 (opt1)</p>	32'hffff_ffff

		<p>S1[x] = 0, S0[x] = 1, alternate function 2 (opt2)</p> <p>S1[x] = 1, S0[x] = 0, alternate function 3 (opt3)</p> <p>S1[x] = 1, S0[x] = 1, multiplexing function 4 (opt4)</p> <p>When [x] = 0, if GPIO_DIR[x] = 0, and GPIO_PULL_EN[x] = 1, the GPIO multiplexing is opt6 analog IO function</p> <p>For the IO multiplexing function, please refer to the chip pin multiplexing relationship</p>	
--	--	--	--

9.4.7 GPIO multiplexing selection register 1

Table 56 GPIOA multiplexing selection register 1

bit access		Instructions	reset value
[15:0]	RW	<p>The high address bit of the GPIO alternate function selection bit, together with GPIO_AF_S0 to determine the alternate function</p> <p>For the IO multiplexing function, please refer to the chip pin multiplexing relationship</p>	16'h0

Table 57 GPIOB multiplexing selection register 1

bit access		Instructions	reset value
[31:0]	RW	<p>The high address bit of the GPIO alternate function selection bit, together with GPIO_AF_S0 to determine the alternate function</p> <p>For the IO multiplexing function, please refer to the chip pin multiplexing relationship</p>	32'h0

9.4.8 GPIO multiplexing selection register 0

Table 58 GPIOA multiplexing selection register 0

bit access		Instructions	reset value
[15:0]	RW	The low address bit of the GPIO alternate function selection bit, and GPIO_AF_S1 together determine the alternate function How to configure see GPIO_AF_SEL register description	16'h0

Table 59 GPIOB multiplexing selection register 0

bit access		Instructions	reset value
[31:0]	RW	The low address bit of the GPIO alternate function selection bit, and GPIO_AF_S1 together determine the alternate function How to configure see GPIO_AF_SEL register description	32'h0

9.4.9 GPIO interrupt trigger mode configuration register

Table 60 GPIOA interrupt trigger mode configuration register

bit access		Instructions	reset value
[15:0]	RW	The interrupt triggering method of GPIO, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] interrupt is edge-triggered [x] = 1, GPIO[x] interrupt is level sensitive	16'h0

Table 61 GPIOB interrupt trigger mode configuration register

bit access		Instructions	reset value
[31:0]	RW	The interrupt triggering method of GPIO, each BIT corresponds to the corresponding GPIO line, 1'bx: [x] = 0, GPIO[x] interrupt is edge-triggered	32'h0

		[x] = 1, GPIO[x] interrupt is level sensitive	
--	--	---	--

9.4.10 GPIO Interrupt Edge Triggered Mode Configuration Register

Table 62 GPIOA Interrupt Edge Triggered Mode Configuration Register

bit access		Instructions	reset value
[15:0]	RW	<p>GPIO interrupt edge trigger mode, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, GPIO[x] edge-triggered interrupt mode is determined by GPIO_IEV</p> <p>[x] = 1, both edges of GPIO[x] trigger an interrupt</p>	16'h0

Table 63 GPIOB Interrupt Edge Triggered Mode Configuration Register

bit access		Instructions	reset value
[31:0]	RW	<p>GPIO interrupt edge trigger mode, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, GPIO[x] edge-triggered interrupt mode is determined by GPIO_IEV</p> <p>[x] = 1, both edges of GPIO[x] trigger an interrupt</p>	32'h0

9.4.11 GPIO interrupt upper and lower edge trigger configuration register

Table 64 GPIOA interrupt upper and lower edge trigger configuration register

bit access		Instructions	reset value
[15:0]	RW	<p>GPIO interrupt upper and lower edge trigger or high and low level trigger selection, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, GPIO[x] interrupt is low level or falling edge triggered</p> <p>[x] = 1, GPIO[x] interrupt is high or rising edge triggered</p>	16'h0

Table 65 GPIOB interrupt upper and lower edge trigger configuration register

bit access		Instructions	reset value
[31:0]	RW	<p>GPIO interrupt upper and lower edge trigger or high and low level trigger selection, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, GPIO[x] interrupt is low level or falling edge triggered</p> <p>[x] = 1, GPIO[x] interrupt is high or rising edge triggered</p>	32'h0

9.4.12 GPIO Interrupt Enable Configuration Register

Table 66 GPIOA Interrupt Enable Configuration Register

bit access		Instructions	reset value
[15:0]	RW	<p>GPIO interrupt enable control, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, GPIO[x] interrupt disabled</p> <p>[x] = 1, GPIO[x] interrupt enable</p>	16'h0

Table 67 GPIOB Interrupt Enable Configuration Register

bit access		Instructions	reset value
[31:0]	RW	<p>GPIO interrupt enable control, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, GPIO[x] interrupt disabled</p> <p>[x] = 1, GPIO[x] interrupt enable</p>	32'h0

9.4.13 GPIO Raw Interrupt Status Register

Table 68 GPIOA Raw Interrupt Status Register

bit access		Instructions	reset value
[15:0]	RW	<p>GPIO bare interrupt status (before MASK), each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, no interrupt is generated for GPIO[x]</p> <p>[x] = 1, GPIO[x] has an interrupt</p>	16'h0

Table 69 GPIOB Raw Interrupt Status Register

bit access		Instructions	reset value
[31:0]	RW	<p>GPIO bare interrupt status (before MASK), each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, no interrupt is generated for GPIO[x]</p> <p>[x] = 1, GPIO[x] has an interrupt</p>	32'h0

9.4.14 GPIO Masked Interrupt Status Register

Table 70 GPIOA Masked Interrupt Status Register

bit access		Instructions	reset value
[15:0]	RW	<p>Interrupt status after GPIO masking (after MASK), each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, no interrupt is generated for GPIO[x] (after MASK)</p> <p>[x] = 1, GPIO[x] interrupt is generated (after MASK)</p>	16'h0

Table 71 GPIOB Masked Interrupt Status Register

bit access		Instructions	reset value
[31:0]	RW	<p>Interrupt status after GPIO masking (after MASK), each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, no interrupt is generated for GPIO[x] (after MASK)</p> <p>[x] = 1, GPIO[x] interrupt is generated (after MASK)</p>	32'h0

9.4.15 GPIO Interrupt Clear Control Register

Table 72 GPIOA Interrupt Clear Control Register

bit access		Instructions	reset value
[15:0]	RW	<p>GPIO interrupt clear control, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, no action</p> <p>[x] = 1, clear GPIO[x] interrupt status</p>	16'h0

Table 73 GPIOB Interrupt Clear Control Register

bit access		Instructions	reset value
[31:0]	RW	<p>GPIO interrupt clear control, each BIT corresponds to the corresponding GPIO line, 1'bx:</p> <p>[x] = 0, no action</p> <p>[x] = 1, clear GPIO[x] interrupt status</p>	32'h0

10 High Speed SPI Device Controller

10.1 Function overview

Compatible with the general SPI physical layer protocol, by agreeing on the data format for interaction with the host, the host can access the device at high speed, up to

The working frequency is 50MHZ.

10.2 Main Features

- ÿ Compatible with general SPI protocol
- ÿ Selectable level interrupt signal
- ÿ Support up to 50Mbps rate
- ÿ Simple frame format, full hardware parsing and DMA

10.3 Functional Description

10.3.1 Introduction to SPI Protocol

SPI works in master-slave mode, usually there is a master device and one or more slave devices, requiring at least 4 wires, in fact 3 wires are also possible (single when transferring). They are SDI (data input), SDO (data output), SCLK (clock), CS (chip select).

- (1) SDI – Serial Data In, serial data input
- (2) SDO – Serial Data Out, serial data output
- (3) SCLK – Serial Clock, clock signal, generated by the master device
- (4) CS – Chip Select, the slave device enable signal, controlled by the master device.

Among them, CS is the control signal of whether the slave chip is selected by the master chip, that is to say, only when the chip select signal is the predetermined enable signal (high

potential or low potential), the operation of the master chip is valid for this slave chip. This makes it possible to connect multiple SPI devices on the same bus.

In addition to the above four signal lines, HSPI also adds an INT line, which generates a drop when the slave device has data to upload.

The interruption of the edge realizes the active reporting of data.

SPI communication is accomplished through data exchange, the data is transmitted bit by bit, the clock pulse is provided by SCLK, SDI, SDO are based on

This pulse completes the data transfer. The data output goes through the SDO line, the data changes on the rising or falling edge of the clock, and the data changes on the following falling edge or

rising edge is read. To complete one-bit data transmission, the same principle is used for input. Therefore, at least 8 changes of the clock signal (the rising edge

and the lower edge is once), to complete the transmission of 8-bit data.

The SCLK signal line is controlled by the master device, and the slave device cannot control the signal line. In an SPI-based device, there is at least one master device.

10.3.2 SPI working process

The HSPI inside the chip works with the wrapper controller. The wrapper controller integrates DMA and is implemented through DMA.

Data exchange between HSPI internal FIFO and chip internal buffer. This operation is implemented in hardware, and software does not need to care about data transmission and reception

In the process, you only need to configure the sending and receiving data linked list, and operate the corresponding registers of the wrapper controller.

For a detailed introduction to the wrapper controller, please refer to the relevant chapters.

10.4 Register Description

10.4.1 Register List of Internal Operation of HSPI Chip

Table 74 HSPI Internal Access Registers

offset address	name	abbreviation	access	describe	reset value
----------------	------	--------------	--------	----------	-------------

0X0000	HSPI FIFO clear register CLEAR_FIFO		RW	clears the contents of the Tx and Rx FIFOs, while A circuit that synchronously resets the system clock domain	0X0000_0000
0X0004	HSPI Configuration Register	SPI_CFG	RW	configures the transmission mode and endianness of SPI set	0X0000_0000
0X0008	HSPI Mode Configuration Register MODE_CFG		RW	Configure ahb master to access the bus burst length	0X0000_0000
0X000C	HSPI Interrupt Configuration Register	SPI_INT_CPU_ MASK	RW	configuration interrupt is enabled	0X0000_0003
0X0010	HSPI Interrupt Status Register	SPI_INT_CPU_ STTS	RW	Get and clear interrupt status	0X0000_0000
0X0018	HSPI data upload length register RX_DAT_LEN	RW	RW	Configure the length of data that can be uploaded	0X0000_0000

10.4.1.1 HSPI FIFO clear register

Table 75 HSPI FIFO clear register

bit access		Instructions	reset value
[31:1] RO		reserve	
[0]	RW	<p>Clear FIFOs, clear the contents of the Tx and Rx FIFOs, and synchronously reset the circuit of the system clock domain (this Except for the registers in the list)</p> <p>0: Do not clear the FIFO 1: Clear valid</p> <p>Set by software, cleared by hardware</p> <p>Note: If you want to reset the whole circuit, you need to use the asynchronous reset leg of this module: rst_n</p>	1'b0

10.4.1.2 HSPI Configuration Register

Table 76 HSPI configuration registers

bit access		Instructions	reset value
[31:4] RO		reserve	
[3]	RW	Bigendian, spi interface supports big and small endian selection of data. 0: support small segment data transfer 1: Support big-endian data transmission	1'b0
[2]	RW	spi_tx_always_drive 0: The spi output is only valid when the chip select is valid, and it is high impedance at other times 1: spi output is always valid	1'b0
[1]	RW	SPI CPHA 0: Transmission mode A 1: Transmission mode B	1'b0
[0]	RW	SPI CPOL, SCK polarity at IDLE 0: 0 when SCK IDLE 1: 1 when SCK IDLE	1'b0

10.4.1.3 HSPI Mode Configuration Register

Table 77 HSPI Mode Configuration Register

bit access		Instructions	reset value
[31:1] RO		reserve	

[0]	RW	Burst len, the burst length when the ahb master accesses the bus 0: burst len is 1 word 1: burst len is 4 characters It is recommended to set the burst transmission of 4 words, so that when the frequency of the spi interface is high, the continuous flow can be guaranteed.	1'b0
-------	----	---	------

10.4.1.4 HSPI Interrupt Configuration Register

Table 78 HSPI Interrupt Configuration Register

bit access		Instructions	reset value
[31:2] RO		reserve	
[1]	RW	IntEnRxOverrun, RxOverrun interrupt enable 0: Rx FIFO overflow interrupt enable 1: Rx FIFO overflow interrupt disabled	1'b1
[0]	RW	IntEnTxUnderrun, TxUnderrun interrupt enable 0: Tx FIFO underflow interrupt enable 1: Tx FIFO underflow interrupt disabled	1'b1

10.4.1.5 HSPI Interrupt Status Register

Table 79 HSPI Interrupt Status Register

bit access		Instructions	reset value
[31:2] RO		reserve	
[1]	RW	RxOverrun 0: Rx FIFO overflow	1'b0

		1: Rx FIFO overflow Write 1 to clear	
[0]	RW	TxUnderrun 0: Tx FIFO underflow 1: Tx FIFO underflow Write 1 to clear	1'b0

10.4.1.6 HSPI data upload length register

Table 80 HSPI data upload length register

bit access		Instructions	reset value
[31:16] RO		reserve	
[15:0] RW		Rx_dat_len Indicates the length of data that can be uploaded, in bytes The upload length is an integer multiple of words. If the upload length is less than a whole word, it will be rounded up.	16'h0

10.4.2 Host side access HSPI controller register list

The host side accesses the SPI interface registers through a fixed SPI command format. The command length is fixed to one byte, and the data length is fixed to two bytes.

Table 81 HSPI Interface Configuration Register (Master Access)

offset address	name	abbreviation	access	describe	reset value

0X02	Get data length register	RX_DAT_LEN	RO	When uploading data, the spi host is used to obtain data from The length of the data read by the device side	0X0000
0X03	Send data flag register	TX_BUFF_AVAIL	RO	When the master sends data to the slave, it is used to judge whether it can be Download data or commands	0X0000
0X04	Reserved	RSV	RO		
0X05	Interrupt configuration register	SPI_INT_HOST_MASK RW	whether to mask the interrupt		0X0000
0X06	Interrupt Status Register	SPI_INT_HOST_STTS	RO	Interrupt status register, the spi host polls this bit Check if there is data to upload	0X0000
0X07	Reserved	RSV	RO		
0X08	Data port 0	DAT_PORT0	RW	The Spi master communicates to the slave device through this register port To send data, the previous data frame is sent using this port	
0X10	Data port 1	DAT_PORT1	RW	The Spi master communicates to the slave device through this register port To send data, send the last data frame to use the port	
0X01	Command port 0	DN_CMD_PORT0	WO	The Spi host sends the slave device through this register Command data, use the previous command data the port	
0X11	Command port 1	DN_CMD_PORT1	WO	The Spi host sends the slave device through this register Command data, send the last frame of command data to make use this port	

10.4.2.1 HSPI get data length register

Table 82 HSPI get data length register

bit	access	Instructions	reset value
[15:0] RO		<p>spi host read-only register, when uploading data, it is mainly used to know how much data is read from the device side</p> <p>But in this module, the upload length is an integer multiple of words. If the upload length value is not an integer word, the host will read</p> <p>Round up when counting, that is, read some redundant bytes</p>	16'h0

10.4.2.2 HSPI send data flag register

Table 83 HSPI send data flag register

bit	access	Instructions	reset value
[15:2] RO		reserve	
[1]	RO	<p>tx_cmdbuf_avail</p> <p>Indicates whether the buff of sending cmd is available, if available, the host can send cmd.</p> <p>0: Send buff is not available</p> <p>1: send buff available</p>	1'b0
[0]	RO	<p>tx_buff_avail</p> <p>Indicates whether the sending buff is available, if available, the host can send data.</p> <p>0: Send buff is not available</p> <p>1: send buff available</p>	1'b0

10.4.2.3 HSPI Interrupt Configuration Register

Table 84 HSPI Interrupt Configuration Register

bit	access	Instructions	reset value
[15:1] RO		reserve	
[0]	RO	IntMaskup_dat_cmd_rdy interrupt mask 0: Interrupts are not masked, interrupts can be generated 1: Interrupts are masked Note: It is recommended to use the host's own internal interrupt mask, which can improve efficiency.	1'b0

10.4.2.4 HSPI Interrupt Status Register

Table 85 HSPI Interrupt Status Register

bit	access	Instructions	reset value
[15:1] RO		reserve	
[0]	RO	up_dat_cmd_rdy Status register for generating interrupt to SPI master 0: Data or command not ready 1: Data or command is ready Readable	1'b0

10.4.2.5 HSPI Data Port 0

Table 86 HSPI data port 0

bit	access	Instructions	reset value
	RW	<p>The SPI host transmits data through the register port and the device, and writes data to this register to send the data.</p> <p>Data can be uploaded by reading from this register. If the frame being transmitted requires multiple transmissions to complete</p> <p>If successful, the register port DAT_PORT1 is used for the last transfer, and DAT_PORT0 is used for the others.</p>	

10.4.2.6 HSPI Data Port 1

Table 87 HSPI Data Port 1

bit	access	Instructions	reset value
	RW	<p>The SPI host transmits data through the register port and the device, and writes data to this register to send the data.</p> <p>Data can be uploaded by reading from this register. If the frame being transmitted requires multiple transmissions to complete</p> <p>If successful, the register port DAT_PORT1 is used for the last transfer, and DAT_PORT0 is used for the others.</p>	

10.4.2.7 HSPI Command Port 0

Table 88 HSPI Command Port 0

bit	access	Instructions	reset value
	RW	<p>The SPI host interacts with the device through this register port, and writes data to this register to issue a command.</p> <p>make. If the command being transferred requires multiple transfers to complete, the last transfer takes the register</p> <p>Port DN_CMD_PORT1, others use DN_CMD_PORT0.</p> <p>Note: This window is only used to issue commands negotiated by the driver and firmware.</p>	

10.4.2.8 HSPI Command Port 1

Table 89 HSPI Command Port 1

bit	access	Instructions	reset value
	RW	<p>The SPI host interacts with the device through this register port, and writes data to this register to issue a command.</p> <p>make. If the command being transferred requires multiple transfers to complete, the last transfer takes the register</p> <p>Port DN_CMD_PORT1, others use DN_CMD_PORT0.</p> <p>Note: This window is only used to issue commands negotiated by the driver and firmware.</p>	

10.4.3 High Speed SPI Device Controller Interface Timing

Mainly describe the SPI read and write timing, and how the main SPI and HSPI interact with each other.

10.4.3.1 Data Format

The data format is divided into two parts: command field and data field, as shown in the figure below. The fixed length of the command field is 8 bits, and the length of the data field is based on the access object.

Different, different length, see below for details.

The highest bit of the command field is the read and write flag bit, and the other 7 bits are the address.

ÿ 0 means read data from the following 7bit address

ÿ 1 means write data to the next 7bit address

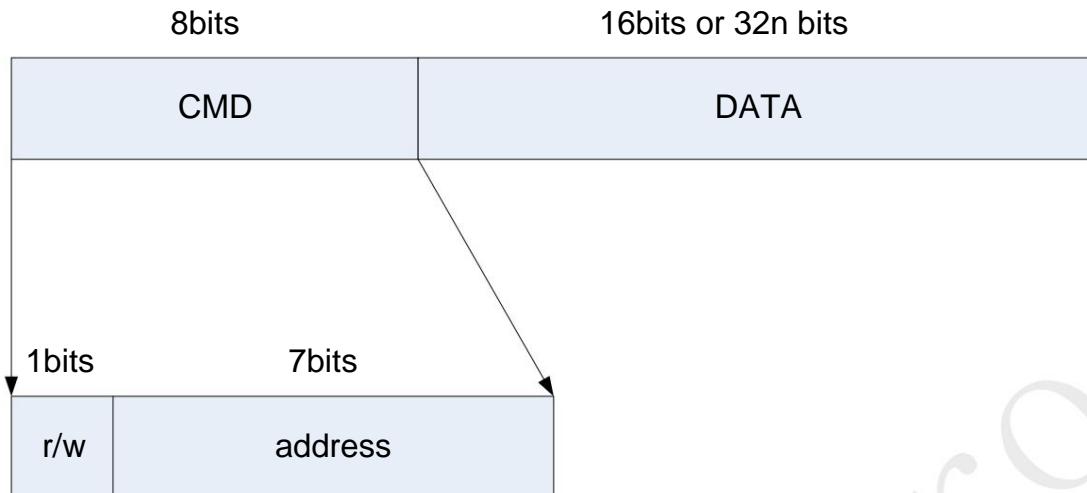


Figure 5 SPI transceiver data format of the host computer

The data field of this module only supports two lengths, the host computer SPI access interface configuration register (Table 2), the data field length is 16bit;

Transfer data through ports (data port 0, data port 1, command port 0 and command port 1), the data field length is an integer of 32 bits

times;

The following figure shows the timing diagram of reading and writing the interface configuration register. The default configuration of the slave device is little endian mode.

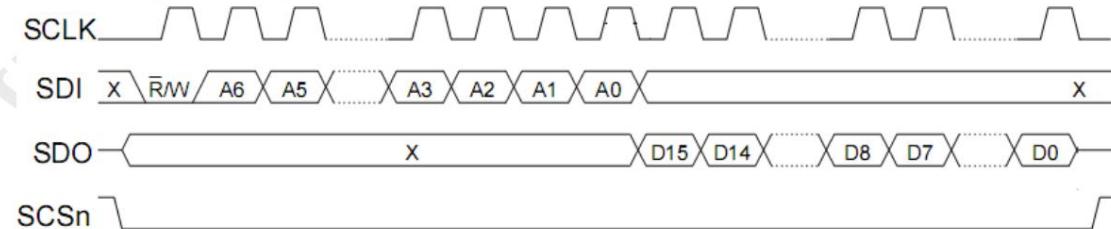


Figure 6 HSPI register read operation (big endian mode)

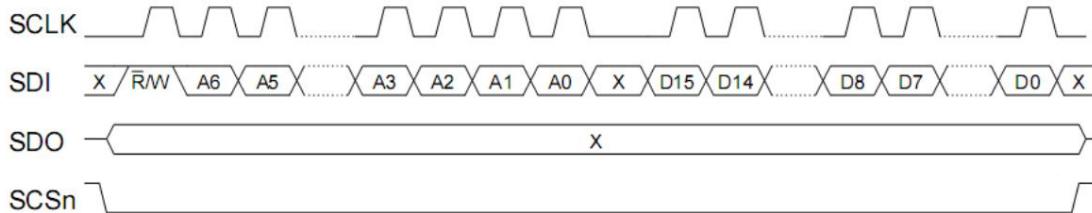


Figure 7 HSPI register write operation (big endian mode)

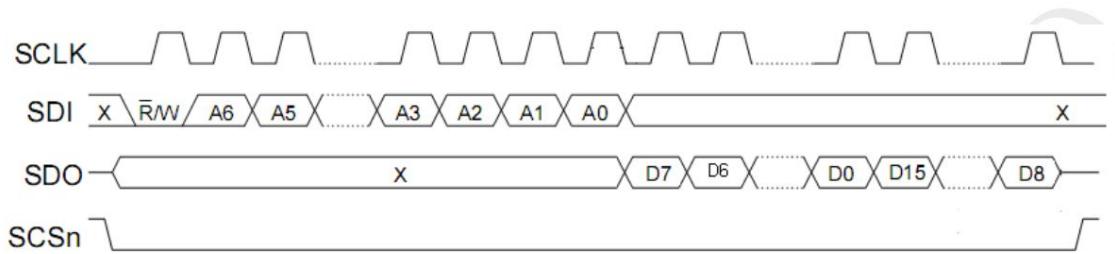


Figure 8 Register read operation (little endian mode)

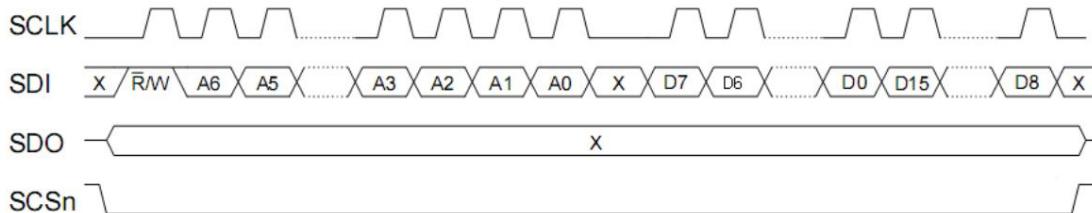


Figure 9 Register write operation (little endian mode)

The following figure is the sequence diagram of reading and writing data, the length of the data field is an integer multiple of 32bit, and the figure only transmits the length of one word.

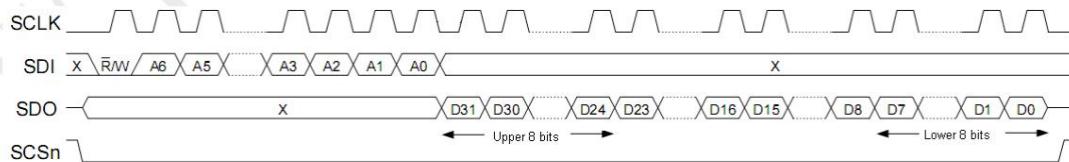


Figure 10 Port read operation (big endian mode)

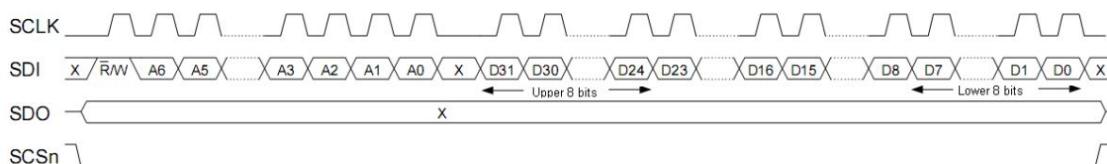


Figure 11 Port write operation (big endian mode)

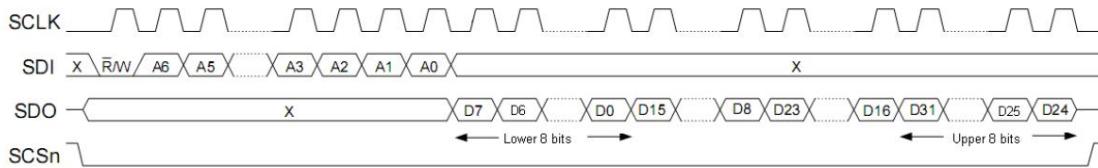


Figure 12 Port read operation (little endian mode)

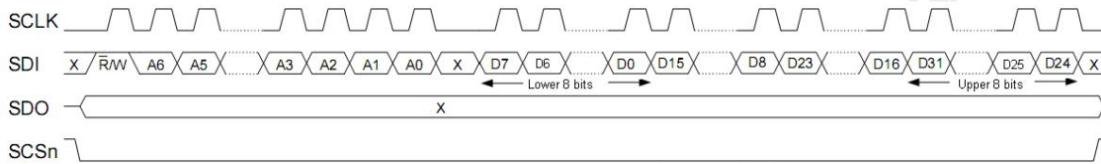


Figure 13 Port write operation (little endian mode)

Note: There can be no waiting time between the command and the data, that is, after the command field is transmitted, the data transmission can be followed, and there is no need for redundant idle clocks or free time. A time delay is fine, but no idle clocks can appear.

10.4.3.2 Timing

This module supports half-duplex, and the supported timings are divided into 4 types according to the clock phase and sampling point. The following timings are given only for clocks

phase and sampling relationship. It should be noted that the chip supports it by default (CPOL=0, CPHA=0).

Note: There can be no waiting time between the command and the data, that is, after the command field is transmitted, the data transmission can be followed, and there is no need for redundant idle clocks or free time. A time delay is fine, but no idle clocks can appear.

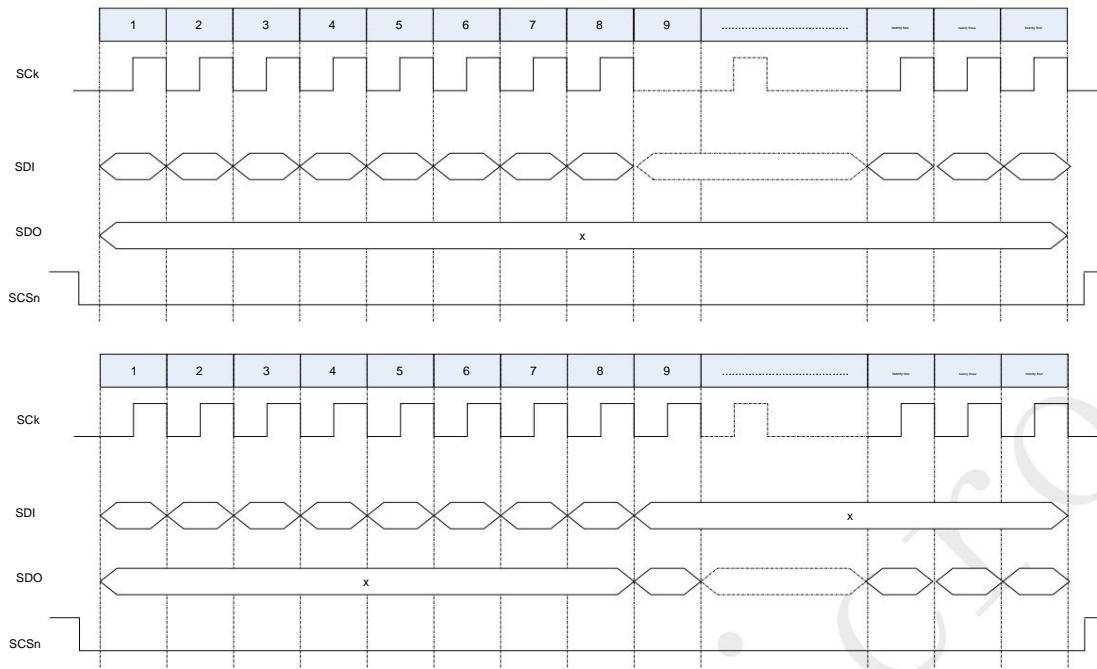


Figure 14 CPOL=0, CPHA=0

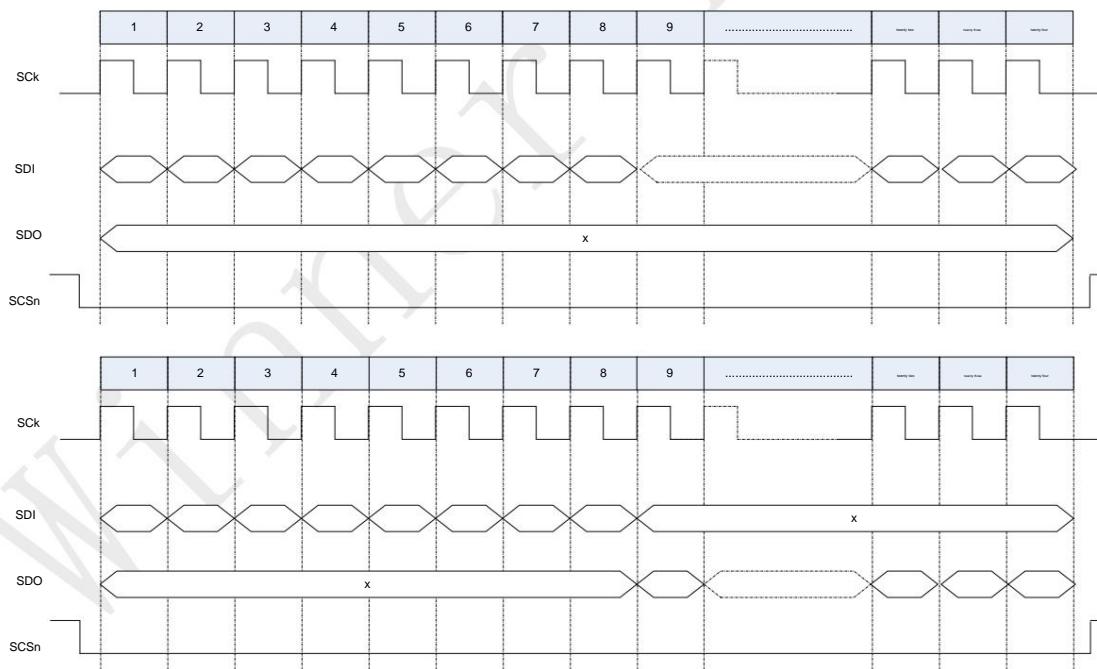


Figure 15 CPOL=0, CPHA=1

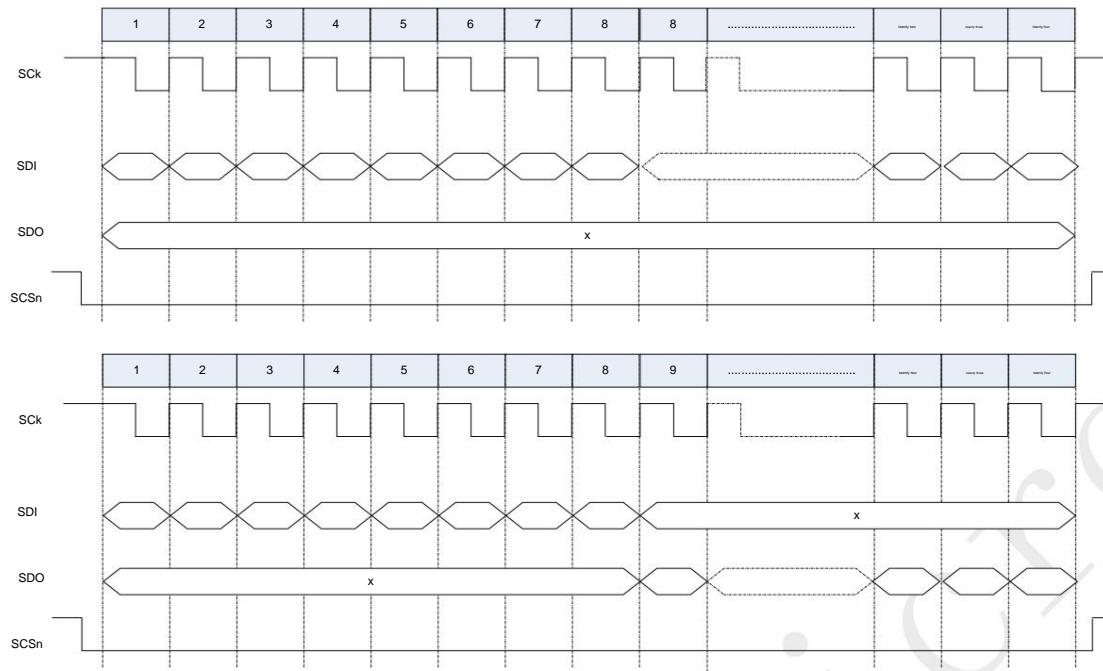


Figure 16 CPOL=1, CPHA=0

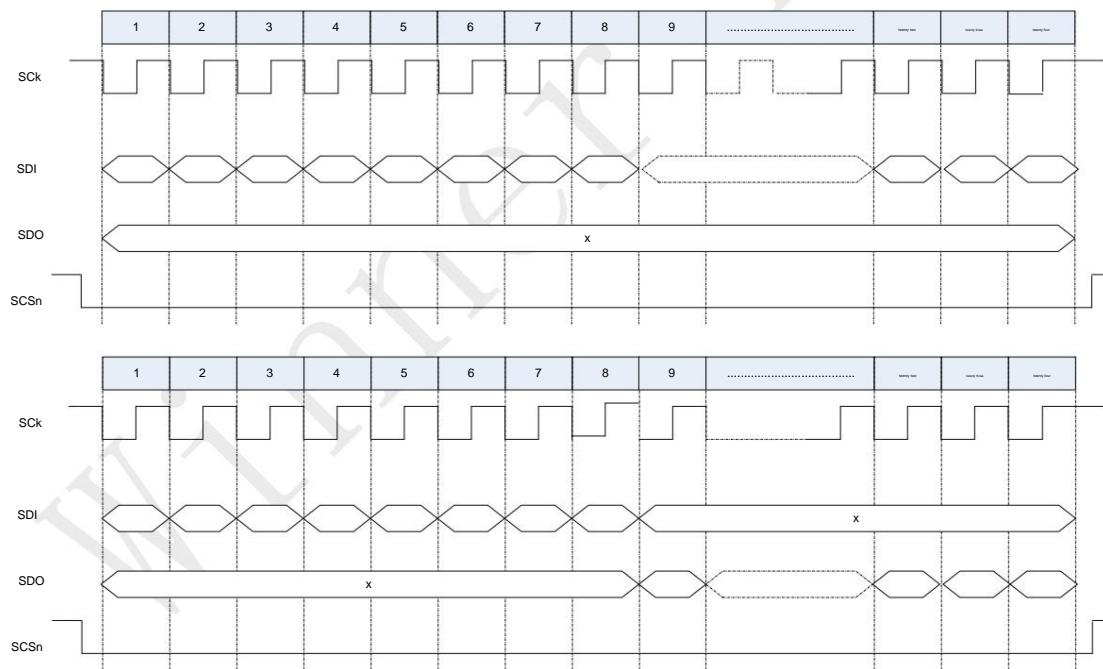


Figure 17 CPOL=1, CPHA=1

10.4.3.3 Interrupts

The interrupt signal is sent by the slave device to the master device, triggered by the SPI_INT pin, active low.

spi_int mainly informs the spi host that there is data or commands that need to be uploaded. The interface registers that the spi host cares about when processing interrupts are:

ÿ SPI_INT_HOST_MASK

ÿ SPI_INT_HOST_STTS

ÿ RX_DAT_LEN

Note: Each uploaded frame corresponds to an interrupt. Only after the frame transmission that needs to be uploaded is completed, if there are still frames to be uploaded, at this time, the

Generate a new interrupt. The diagram below is one way to handle interrupts.

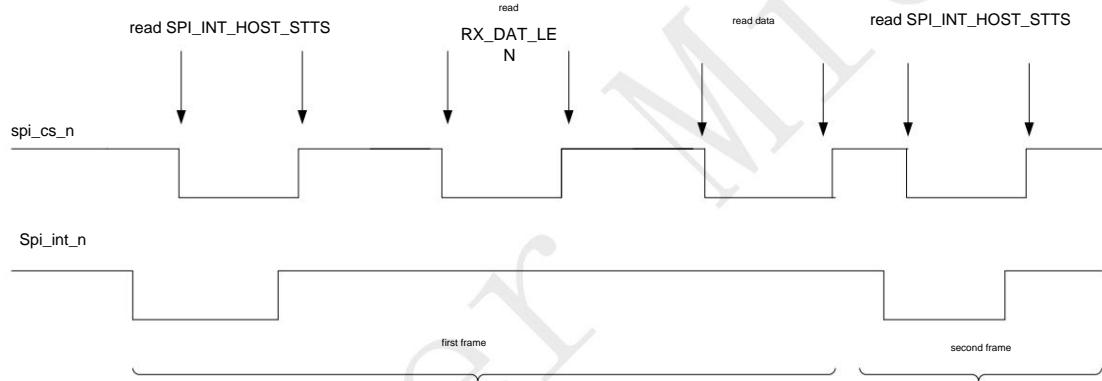


Figure 18 Main SPI processing interrupt flow

10.4.3.4 Main SPI Transceiver Data Workflow

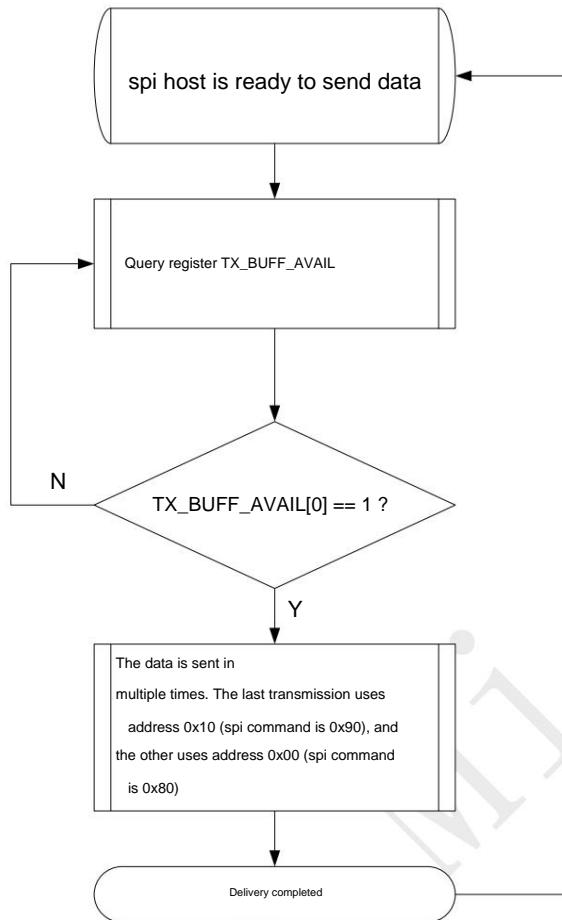


Figure 19 Downlink data flow chart

Note: The length of the data to be sent must be in word units. If it is not a whole word, fill in 0 and complete it.

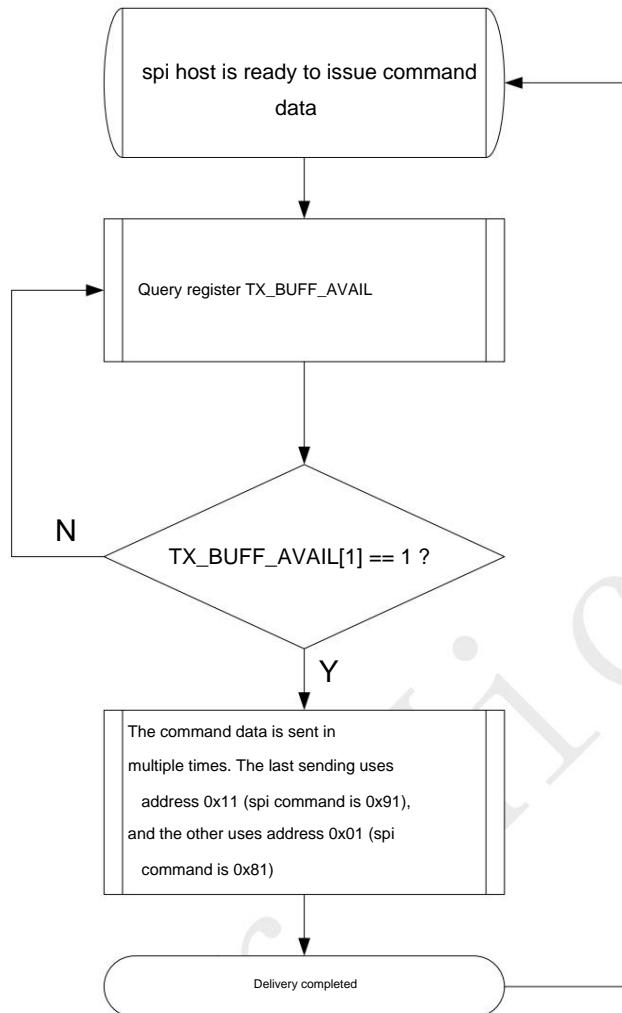


Figure 20 Downlink command flow chart

Note: The length of the issued command must be in word units. If it is not a whole word, fill in 0 and make up for it.

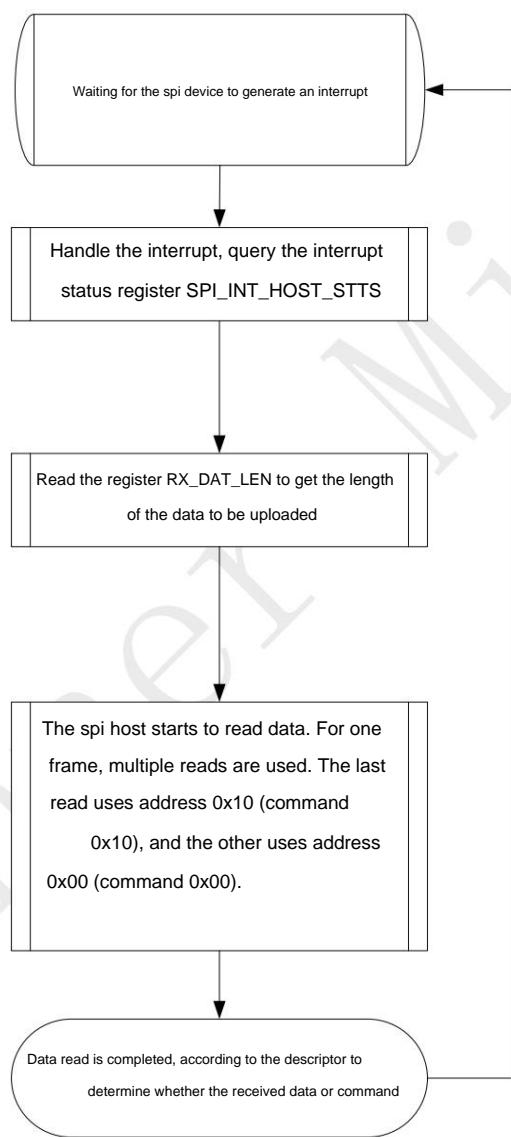


Figure 21 Upstream data (command) flow chart

Upstream data and upstream commands have the same process.

It should be noted here that the length of the upstream data must be in words. If the effective length is not an integer word, the excess data at the tail can be thrown away.

Lose.

It should be noted that there are two channels of data and commands between the master and slave to exchange data, and the user can choose any channel to use

use or both. The maximum data length of one exchange of command channel is 256 bytes, and the maximum length of one exchange of data channel is 256 bytes.

1500 bytes. The data length limit is controlled by the slave device. If the length exceeds the limit, the data structure of the slave device will be destroyed.

11 SDIO Device Controller

11.1 Function overview

W800 integrates the SDIO device interface, as a slave device, to complete the data interaction with the host. Internally integrated 1024byte asynchronous

FIFO, completes the data interaction between the host and the chip.

11.2 Main Features

- ÿ Compatible with SDIO Card Specification 2.0
- ÿ Support host rate 0~50MHz
- ÿ Support blocks up to 1024 bytes
- ÿ Supports 1-bit SD and 4-bit SD modes

11.3 Functional Description

11.3.1 SDIO bus

The SDIO bus is similar to the USB bus. The SDIO bus also has two ends, one of which is the host side and the other side is the device side .

The design of DEVICE is to simplify the design of DEVICE, and all communications are started by commands issued by the HOST side. exist

As long as the DEVICE side can parse the HOST command, it can communicate with the HOST, and the SDIO HOST can connect multiple
DEVICE.

In the SDIO bus definition, the DAT1 signal line is multiplexed as an interrupt line. In the 1BIT mode of SDIO, DAT0 is used to transmit data, DAT1
used as an interrupt line. In the 4BIT mode of SDIO, DAT0 -DAT3 are used to transmit data, and DAT1 is multiplexed as an interrupt line.

11.3.2 SDIO Commands

On the SDIO bus, the HOST side initiates the request, and then the DEVICE side responds to the request. The request and response will contain data information:

ÿ Command: The command used to start the transmission is sent from the HOST side to the DEVICE side, where the command is sent through the CMD message.

transmitted by the number line;

ÿ Response: The response is returned by DEVICE as the response of Command. It is also transmitted through the CMD line;

ÿ Data: Data is transmitted in both directions. Can be set to 1-wire mode or 4-wire mode. Data is passed through DAT0-

DAT3 signal line transmission.

Each operation of SDIO is initiated by HOST on the CMD line to initiate a CMD. For some CMDs, DEVICE needs to return Response,

Some don't.

For the read command, firstly HOST will send a command to DEVICE, and then DEVICE will return a handshake signal. At this time, when HOST

After receiving the response handshake signal, the data will be placed on the 4-bit data line, and the CRC check code will be followed when the data is transmitted. when the whole

After the read transfer is completed, HOST will send a command again to notify DEVICE that the operation is completed, and DEVICE will return a response at the same time.

For the write command, first HOST will send a command to DEVICE, and then DEVICE will return a handshake signal. At this time, when HOST

After receiving the response handshake signal, the data will be placed on the 4-bit data line, and the CRC check code will be followed when the data is transmitted. when the whole

After the write transfer is completed, HOST will send a command again to notify DEVICE that the operation is completed, and DEVICE will return a response at the same time.

11.3.3 SDIO Internal Storage

The SDIO device has a fixed storage map, including the general information area (CIA) and the special function area (function unique area).

The registers in CIA include I/O port function, interrupt generation and port work information, which can be defined by CIA through read and write function 0.

register for related operations. CIA includes CCCR, FBR and CIS three aspects of information. Among them, CCCR defines the public

Common control register, the host side can check the SDIO card and operate the port by operating CCCR. The address of CCCR is 0X00-

0xFF. FBR defines the supported operations of port function 1 to port function 7, including the requirements and functions of each port, power control, etc.

The address of FBR is 0Xn00-0Xnff (where n is the function port number). CIS defines some information structures of the card, the address is 0X1000-

0X17FFF, CIS has a public CIS and its own CIS for each functional port, where the initial address of the public CIS is in the CIS Pointer of CCCR

In the domain, the CIS of each port function is in the CIS Pointer domain of the FBR of each functional port.

The storage map of CIA is as shown below.

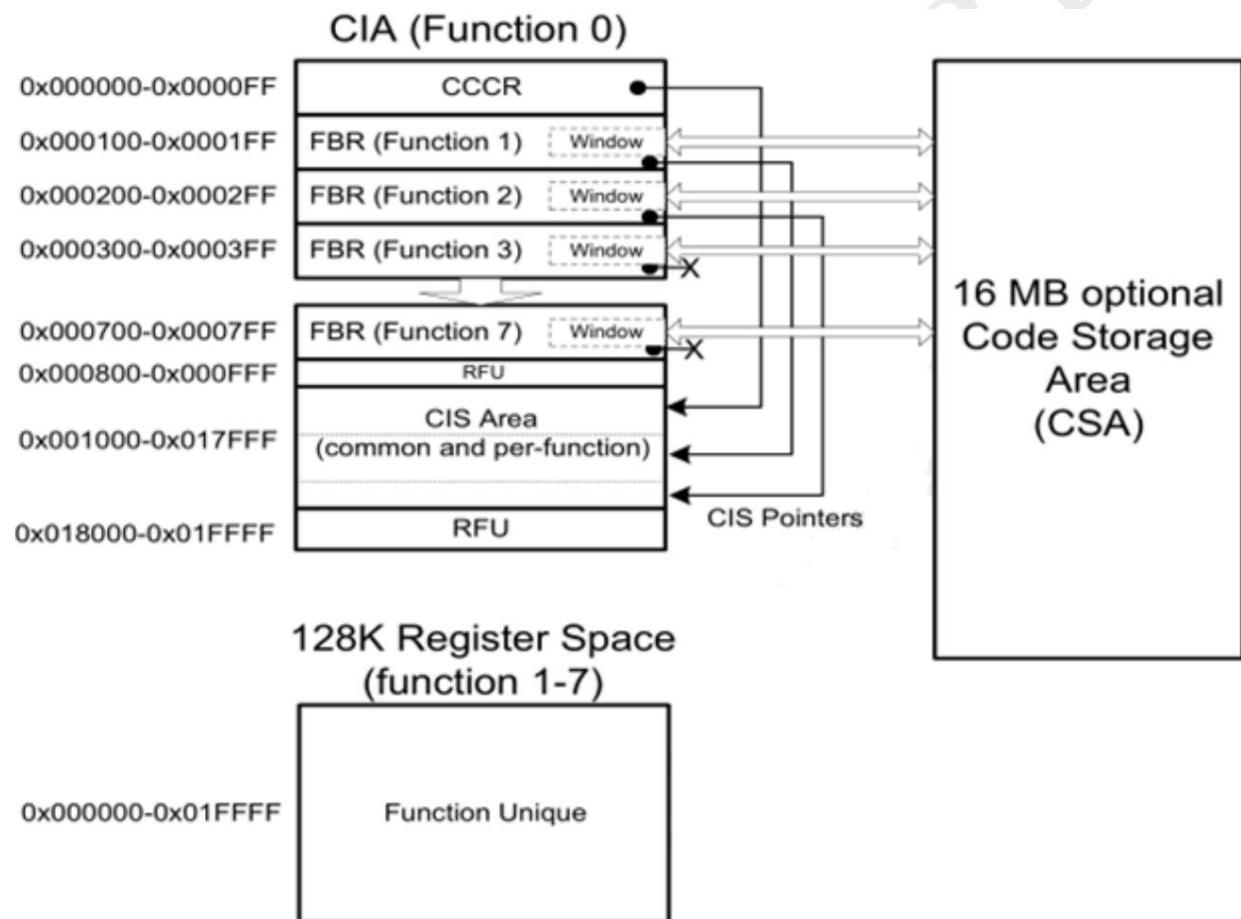


Figure 22 SDIO internal storage map

The description of each register in CIA refers to the following. For an in-depth look at CIA, see the SDIO Protocol Specification.

11.4 Register Description

11.4.1 Register List

11.4.2 SDIO Fn0 register

The Fn0 register is a register specified by the SDIO protocol, and its address range is: 0x00000~0x1FFF, a total of 128K. The starting address is 0x00000.

The Fn0 register is accessed by the SDIO host through the CMD52 command, the offset address is the access address, and the function number is 0.

Address	Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	CCCR/SDIO Revision	SDIO bit 3	SDIO bit 2	SDIO bit 1	SDIO bit 0	CCCR bit 3	CCCR bit 2	CCCR bit 1	CCCR bit 0
0x01	SD Specification Revision	RFU	RFU	RFU	RFU	SD bit 3	SD bit 2	SD bit 1	SD bit 0
0x02	I/O Enable	IOE7	IOE6	IOE5	IOE4	IOE3	IOE2	IOE1	RFU
0x03	I/O Ready	IOR7	IOR6	IOR5	IOR4	IOR3	IOR2	IOR1	RFU
0x04	Int Enable	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IENM
0x05	Int Pending	INT7	INT6	INT5	INT4	INT3	INT2	INT1	RFU
0x06	I/O Abort	RFU	RFU	RFU	RFU	RES	AS2	AS1	AS0
0x07	Bus Interface Control	CD Disable	SCSI	ECSI	RFU	RFU	RFU	Bus Width 1	Bus Width 0
0x08	Card Capability	4BLS	LSC	E4MI	S4MI	SBS	SRW	SMB	SDC
0x09-0x0B	Common CIS Pointer	Pointer to card's common Card Information Structure (CIS)							
0x0C	Bus Suspend	RFU	RFU	RFU	RFU	RFU	RFU	BR	BS
0x0D	Function Select	DF	RFU	RFU	RFU	FS3	FS2	FS1	FS0
0x0E	Exec Flags	EX7	EX6	EX5	EX4	EX3	EX2	EX1	EXM
0x0F	Ready Flags	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RFM
0x10-0x11	FN0 Block Size	I/O block size for Function 0							
0x12	Power Control	Reserved for Future Use (RFU)						EMPC	SMPC
0x13	High-Speed	RFU	RFU	RFU	RFU	RFU	RFU	EHS	SHS
0x14-0xEF	RFU	Reserved for Future Use (RFU)							
0xF0-0xFF	Reserved for Vendors	Area Reserved for Vendor Unique Registers							

Figure 23 CCCR register storage structure

Address	7	6	5	4	3	2	1	0
0x100	Function 1 CSA enable	Function 1 supports CSA	RFU	RFU	Function 1 Standard SDIO Function interface code			
0x101					Function 1 Extended standard SDIO Function interface code			
0x102	RFU	RFU	RFU	RFU	RFU	RFU	EPS	SPS
0x103-0x108					Reserved for Future Use (RFU)			
0x109-0x10B					Pointer to Function 1 Card Information Structure (CIS)			
0x10C-0x10E					Pointer to Function 1 Code Storage Area (CSA)			
0x10F					Data access window to Function 1 Code Storage Area (CSA)			
0x110-0x111					I/O block size for Function 1			
0x112-0x1FF					Reserved for Future Use			
0x200-0x7FF					Function 2 to 7 Function Basic Information Registers (FBR)			
0x800-0xFFFF					Reserved for Future Use			

Figure 24 FBR1 register structure

Address	7	6	5	4	3	2	1	0
0x0001000 - 0x017FFF					Card Common Card Information Structure (CIS) area for card common and all functions			
0x018000-0x01FFFF					Reserved for Future Use			

Figure 25 CIS storage space structure

11.4.2.1 SDIO CCCR register and FBR1 register list

Table 90 SDIO CCCR register and FBR1 register list

offset address	name abbreviated access			describe	reset value
0x00	CCCR/SDI O Revision	SDIOx RO		[3:0], indicates the supported CCCR/FBR format 4'h0: CCCR/FBR Version 1.00 4'h1: CCCR/FBR Version 1.10 4'h2: CCCR/FBR Version 1.20 4'h3-4'hF: Rsv Represented by CIA Register[3:0]	4'h2

		CCCRx RO		[7:4], indicates the supported SDIO protocol version 4'h0: SDIO Version 1.00 4'h1: SDIO Version 1.10 4'h2: SDIO Version 1.20 (unreleased) 4'h3: SDIO Version 2.00 4'h4-4'hF: Rsv Represented by CIA Register[7:4]	4'h3
0X01	SD specification Revision	SDx	RO [3:0], indicates the supported SD protocol version 4'h0: SD Physical Version 1.01 (March 2000) 4'h1: SD Physical Version 1.10 (October 2004) 4'h2: SD Physical Version 2.00 (May 2006) 4'h3-4'hF: Rsv Represented by CIA Register[11:8]	4'h2	
				RFU	4'h0
0X02	I/O Enable IOEx		RO RW [7:1], Function enable, bit1-bit7 correspond to 7 functions respectively, corresponding to If the SD host sets the corresponding bit to 1, the corresponding function is enabled, otherwise the corresponding function does not work. Note: CIS0, CIS1 and CSA are placed in Fn1, even if Fn1 is not enabled at this time, SD host can also read and write to these three areas (CIS0, CIS1 cannot Write).	RFU 7'b0	
0X03	I/O Ready IORx		RO	RFU	1'b0

			RO	[7:1], IOR has a total of 7 bits, corresponding to the status of the 7 functions, if the corresponding bit If it is 1, it means that the function can work. In this design, HC8051 configures the function of program register The ready bit is 1, so that the bit1 of this register is set to 1, so that the flag Fn1 can be normal. Work. Note: Read and write operations to CIS0, CIS1, CSA are independent of IOR1, that is, even if If IOR1=0, the contents of these three storage spaces can also be accessed.	7'b0
0X04 Int Enable IENM RW [0]			interrupt enable signal	0: Interrupts from the card cannot be sent to the SD host 1: Any function interrupt can be sent to the host	1'b0
				IENx RW [7:1], interrupt enable of functionx. If IEN1=0, the interrupt from Fn1 will not be sent to the host. IEN1=1, then the interrupt of Fn1 is allowed to be sent to the host	7'b0
0X05 Int Pending			RO	[0], RFU	1'b0
			INTx	RO [7:1], Interrupt for functionx is pending. INT1=0, interrupts without Fn1 are pending INT1=1, Fn1 has an interrupt pending. Note: If IEN1 and IENM are not 1, the host will not receive pending interrupts	7'b0
0X06 I/O Abort ASx			WO [2:0]	cancel IO read or write, thereby releasing the bus. To cancel the Fn1 operation, CMD52 should be used to write 3'b1. This command is under SPI not support.	3'b0
			RES	WO [3], soft reset signal	1'b0

				1: Reset the circuit of the SD clock domain, this bit is automatically cleared after being set, no special Door clears. This reset signal will not affect the current card protocol selection (SD or SPI mode) without affecting CD Disable. Only use CMD52 operation.	
		RO	RFU		4'b0
0X07 Bus	Interface control	Bus Width	RW [1:0], data line width 2'b00: 1bit data line mode 2'b10: 4bit data line mode Reset or power on, it will become 2'b00		2'b00
		RO	[4:2], RFU		3'b000
		ECSI RW [5]	enable continuous SPI interrupt. If SCSI is 1, this register is used to enable SDIO card in SPI mode In this case, an interrupt is given at any time, and the state of the CS line does not need to be concerned at this time.		1'b0
	SCSI	RO [6]	supports continuous SPI interrupts. If it is 1, it means that the SDIO card supports SPI mode, giving break without caring about the state of the CS. This register is set when program reg[2] is 1.		1'b1
	CD Disable	RW [7]	connect or disconnect the 10-90K pull-up resistor on CD/DAT[3] (pin1). 0: connect pull-up resistor 1: Disconnect the pull-up resistor After power-up, this register is cleared, that is, a pull-up resistor is connected. The state of this register is not Will be affected by the reset command in the SD protocol.		1'b0
0X08 Card		SDC	RO[0], supports the execution of CMD52 commands during data transfer. 1'b1 is not supported in SPI mode		

Capability			this register. When program reg[3] is 1, this register is 1	
	SMB	RO	[1], means that the SDIO card supports the Block transfer mode required by CMD53. When program_reg[4] is 1, this register is 1	1'b1
	SRW	RO	[2], indicates that SDIO supports read wait - Read Wait Control (RWC) operation. When program_reg[5] is 1, this register is 1	1'b1
	SBS	RO	[3] . means that the SDIO card supports suspend/resume. If 0, (0x0C-0x0F) registers are not supported If 1, except Fn0, all functions will be suspended according to SD host requirements or restore When program_reg[6] is 1, this register is 1	1'b1
	S4MI	RO	[4], indicates that the SDIO card supports the 4bit multi-block data transmission mode to the host An interrupt is generated. 0: Interrupts between block transfers are not supported, in this case, as long as IENx=1, SDIO can still initiate an interrupt to the host in other interrupt cycles 1: Support to generate interrupts between block transfers When program_reg[7] is 1, this register is 1	1'b1
	E4MI RW [5], interrupt enable.		Allow 4bit multi-block mode, in the middle of two block data transmission to the host Interrupted. 0: not allowed 1: Allow	1'b0

				A power-on reset or reset command will clear this register to 0	
	LSC	RO	[6], 0: Indicates that the SDIO card is a full-speed device 1: Indicates that the SDIO card is a low-speed device When program_reg[8] is 1, this register is 1	1'b0	
	4BLS	RO	[7], 0: Indicates that SDIO is a low-speed mode device or does not support 4bit mode 1: Indicates that SDIO is a low-speed mode device and supports 4bit mode When program_reg[9] is 1, this register is 1	1'b1	
0X09- 0X0B	Common CIS pointer RO		[23:0], points to the starting address pointer of SDIO card shared CIS (CIS0). CIS0 package Contains information about the entire card. Its access space is: 0x001000-0x017FFF. Pointers are stored in little endian format (LSB).	24'h00 1000	
0X0C Bus	Suspend	BS	RO [0], bus state. 0: The currently selected function does not use the data bus 1: The currently selected function (using FSx or using the function in the IO command number) is executing the command that will transmit data on the data line This register is used by the host to determine which function is currently using the data total Wire. If the SDIO card does not support the suspend-resume function, this register is 0. Any access to the CIA cannot be suspended, this register is always 1, even if the BR register is 1.	1'b0	

				In SPI mode, read-only, and is 0.	
		BR	RW	<p>bus release request/status. This register is used to request the selected function (with FSx or CMD53 total function number selected) release the data bus and hang up related operations.</p> <p>If the host sets this register to 1, the selected function will temporarily stop.</p> <p>According to the data transmission on the line, and suspend the command of the current data operation. BR register keeps</p> <p>Holds at 1 until the release process is complete. Once the function is suspended, the device clears the</p> <p>Zero BS, BR to notify the host. The host can monitor pending requests by reading the BR</p> <p>Execution status, if BR is 1, the pending request is still executing. The host can actively</p> <p>Write 0 to BR to cancel an executing pending request.</p> <p>In SPI mode, read-only, and is 0.</p>	4'h0
		RO	[7:2], RFU		6'b0
0X0D Function	Select	FSx	RW[3:0]	<p>used to select function[0-7] in suspend/resume operation. two methods</p> <p>Write FSx:</p> <p>IO write operation to CCCR</p> <p>A newly initiated IO command will cause FSx to be set to the function in the command number.</p> <p>If the function is currently suspended, write the function's number, when reading FSx, the data transfer operation of this function will be resumed.</p> <p>do. The returned value will be the number of the currently selected function.</p> <p>Note: When reading FSx, if BS=0, the value of FSx is undefined.</p>	4'b0

				4'b0000: Transaction of function 0 (CIA) 4'b0001-4'b0111: Transaction to functions 1-7 4'b1000: Transaction of memory in combo card 4'b1001-4'b1111: Not defined, reserved for future use	
		RO	[3:1], RFU		3'b000
	DF			RO[7], restore data flag. Writing the function number to FSx will restore the selection Data transfer in function. Once data transfer resumes, the DF register will indicate Is there more data to transfer. 0: No more data to transfer after the function is resumed. 1: After the function is resumed, there is more data to transfer. DF is used to control the interrupt period in 4bit mode. If 1, in function After recovery, there is more data to transfer, in which case the interruption period is canceled. If 0, the function resumes after the end of the data transfer (in the busy case), In this case, after recovery, there is no data transfer, so the host can The start of the interrupt period is detected after the function resumes.	1'b0
0X0E Exec Flags EXx			RO[7:0]	execute flag. The host determines all functions through these bits [7- 1] The status of executing the command. These registers tell the host that a function is Execute the command, so no new commands can be issued for this function. In SPI mode, read-only, and is 0.	8'h00
0X0F Ready	Flags	RFx	RO [7:0]	read flag. The host can know the function[7- 1] Read and write busy state. If a function is executing a write transaction, the corresponding	8'h00

				The RFx bit is cleared to indicate that the function is busy and not ready to receive more data according to. If a function is performing a read operation, the corresponding RFx bit is cleared if it is zero, it indicates that the read data is invalid, and if it is 1, it indicates that the read data can be transmitted. SPI has no effect, read only, and is 0	
0X10- 0X11	FN0 Block Size	RW [15:0]		Block size when the block corresponding to Fn0 is transmitted. maximum 2048Byte, minimum 1Byte. The storage method is small segment format (LSB)	16'h00
0X12 Power Control	SMPC RO [0], supports host power control.			0: SDIO total current is less than 200mA, even if all functions are valid (IOEx=1). EMPC, SPS, EPS are all 0. 1: The total SDIO current can exceed 200mA. EMPC, SPS, EPS are valid.	1'b1
	EMPC RO [1], host power control enable.			0: The total current of SDIO card is less than 200mA. SDIO card auto switch function(s) to low current mode or not allow some functions to be enabled, and it ignores the EPS value, so that the current of the card is less than or equal to 200mA. 1: The total current of SDIO card can exceed 200mA, and SPS and EPS are valid. host Use the SPS, EPS and IOEx in the FBR according to their ability to provide current. function of higher current.	1'b0
		RO	[7:2], RFU		
0X13 High Speed	SHS	RO		[0], indicates that the SDIO card supports high-speed 0: High speed not supported 1: support high speed	1'b1

		EHS	RW [1], High speed enable	0: SDIO card works at the default speed, the highest frequency is 25MHZ 1: SDIO card can work in high-speed mode, the highest frequency is 50MHZ	1'b0
		RO	[7-2], RFU		
0X14-	RFU	RO	Reserved for Future Use (RFU)		8'b0
0XEF					
0XF0-	Reserved for	RO	Area Reserved for Vendor Unique Registers		8'b0
0xFF	Vendors				
0X100 I/O	Device Interface	RO	[3:0], what kind of device is the flag Fn1.		4'h7
	Code		Programmable by register CIA[15:12].		
			4'h0 No SDIO standard interface supported by this function		
			4'h1 This function supports the SDIO Standard UART		
			4'h2 This function supports the SDIO Type-A for Bluetooth		
			standard interface		
			4'h3 This function supports the SDIO Type-B for Bluetooth		
			standard interface		
			4'h4 This function supports the SDIO GPS standard		
			interface		
			4'h5 This function supports the SDIO Camera standard		
			interface		
			4'h6 This function supports the SDIO PHS standard		
			interface		

			4'h7 This function supports the SDIO WLAN interface 4'h8 This function supports the Embedded SDIO-ATA standard interface	
RFU	RO	[5:4], RFU		2'b00
Function supports CSA	RO	[6], 0: Fn1 does not support CSA 1: Fn1 support with CSA Can be programmed through register CIA[16]		1'b0
Function CSA enable	RW[7],	0: Access to CSA is not allowed 1: Allow access to CSA		1'b0
0X101 Extended standard I/O device type code	RO	[7:0], extension of I/O Device Interface Code can be programmed through registers CIA[24:17]		8'b0
0X102 SPS	RO	[0], indicates whether Fn1 has power consumption selection 0: No power consumption option 1: There are two power consumption options, which can be selected by EPS can be programmed through register CIA[25]		1'b0
EPS	RW [1], power consumption selection	0: Fn1 works in high current mode 1: Fn1 works in low current mode		1'b0

		RO	[7:2], RFU	6'b0
0X103- 0X108		RO	RFU	0
0X109- 0X10B	Address pointer to function CIS1	RO	[16:0], The CIS address pointer of Fn1, namely CIS1, instructs the host to access the CIS of Fn1 initial address. The storage method is LSB segment format.	17'h02 000
		RO	[23:17], RFU	7'b0
		RW [23:0]	points to the 24bit address pointer of CSA, the host accesses through the CSA access window After asking CSA, the pointer is automatically incremented by 1. Addresses are stored in small segment format (LSB)	24'h00 0000
0X10F	Data access window to CSA	RW [7:0]	read and write window to CSA. When writing to this address, the corresponding data will be written to the address indicated by the CSA 24bit address pointer through this window, During read operation, the data is read from the address indicated by the 24bit CSA address pointer. This window is sent to the host.	8'b00
0X110- 0X111	Function1 IO Block Size	RW [15:0]	16bit register, used to set the IO block size. biggest The block size is 2048Byte, and the minimum is 1. This data is stored in little endian mode (LSB).	16'b0
0X100- 0X101- 0	CIS0	The RO host	accesses the CIS address space of Fn0, that is, the host accesses through this address space segment CIS0. This SDIO card supports up to 17 bytes of CIS0	
0X200- 0-	CIS1	The RO host	accesses the CIS address space of Fn1, that is, the host accesses through this address space segment CIS1. This SDIO card supports CIS1 byte data of 55~308.	

0X213				
3				
RFU			RFU	

11.4.3 SDIO Fn1 register

The Fn1 register is the address space allocated to function1 by the SDIO protocol, and its address range is: 0x00000~0x1FFFF, a total of 128K.

Since the internal AHB bus address bit width of the chip is 32 bits, SDIO cannot use the 17-bit address to directly access the chip.

In the design, one address mapping needs to be completed. The specific mapping relationship is as follows: (FN1 access space)

Table 91 SDIO Fn1 address mapping relationship

SDIO host access window	Corresponds to the actual physical address space	Actual physical address space contents
0X0000 ~ 0X00FF	0X0000 ~ 0X00FF	SDIO module internal register address space.
0X1000 ~ 0X1FFF	Configurable	CIS0 physical space, the specific physical space is configured by firmware.
0X2000 ~ 0X2FFF	Configurable	CIS1 physical space, the specific physical space is configured by firmware.
0X4000 ~ 0X4FFF	Configurable	Downlink and uplink cmd physical space, specific physical space The address is configured by firmware.
0X5000 ~ 0X5FFF 0X15000 ~ 0X15FFF	variable	Send buffer address space, according to sdio_txbd instructions in .
0X6000 ~ 0X7FFF 0X16000 ~ 0X17FFF	variable	Receive buffer address space, according to sdio_rxbd instructions in .
0X8000 ~ 0X9FFF	0X0E000000 ~ 0X0E002000	AHB bus config address space.
0XA000 ~ 0XBFFF	0X0F000000 ~ 0X0F002000	AHB bus APB address space.

Drivers should avoid accessing spaces beyond the above range, as doing so may have unexpected results.

The first address space register is inside SDIO and can only be accessed by SDIO HOST; access to other address spaces will be based on

The description maps to other spaces inside the chip.

This section only introduces the registers in the SDIO 0x0000 ~ 0x00FF address space, which are commanded by the SDIO host through CMD52

For direct access, the offset address is the access address, and the function number is 1.

Table 92 SDIO Fn1 part register (for HOST access)

offset address	name	Bit wide	access	description	reset value
0X00~0X03			RO	RSV	
0X04	int_read_data	[7:1]	RO	RSV	7'b0
		[0]	RW	Upstream data interruption. Active high, write 1 to clear. When 0x1C is read, it will also be automatically cleared to 0.	1'b0
0X05	int_mask	[7:1]	RO	RSV	7'b0
		[0]	RW	corresponds to the mask enable signal of int_src. 1 to mask the corresponding interrupt. 1'd0	1'b0
0X06	wlan_awake_stts[0]	[7:2]	RO	RSV	6'b0
			RO	Current WLAN status: 1 is ACTIVE; 0 is SLEEP.	1'b1
0X1C	dat_len0	[6:0]	RO	Upstream data length is 7 bits high. A total of 12 bits, the lower 5 bits are at 0x1D middle.	7'b0
	dat_vld	[7]	RO	1'b1	1'b1

0X1D	dat_len1	[7:3]	RO Upstream	data length is 5 bits lower. A total of 12 bits, the high 7 bits are at 0x1C middle.	5'b0
		[2:0]	RO		3'b0
0X1E			RO	RSV	
0X1F		[7:2]	RO	RSV	6'b0
	down_cmdbuf_vl	[1]	RO down	link command buffer is available, 1 is valid.	1'b1
	d				
	txbuf_vld	[0]	RO down	stream data buffer is available. 1 is valid, indicating that there is an available send buffer.	1'b0
0X20	wlan_wake_en	[0]	In RW SLEEP	state, the chip wake-up sent by SDIO is enabled, active high. After the chip is woken up, this bit will be automatically cleared to 0 by hardware.	1'b0
0X21		[7:1]	RO	RSV	7'b0
	fn1_RST	[0]	RW Soft	reset, 1 is valid. After the software writes 1, (ie the wlan part of the chip circuit) is reset, write 0 After that, function1 reset is released.	1'b0
0X22		[7:1]	RO	RSV	7'b0
	fn1_recov	[0]	RW Error	recovery enable, 1 is valid, after the command response ends, the The bit is automatically cleared to 0. This function is used to complete the same function as fn0/fn1 io abort. This register can be set when cmd is abnormal or the command is terminated prematurely 1, to complete the io abort operation. Because in some bus driver versions, the io abort command is restricted	1'b0

				(that is, the access address space of this register is limited), user drivers are not allowed to use At this time, writing 1 to this register can replace the io abort operation, and produce the same effect.	
--	--	--	--	--	--

11.4.3.1 SDIO AHB Interface Slave Register

The following registers are used when the SDIO slave device is initialized.

When transferring data, it needs to be used in conjunction with the wrapper controller. For the part of the wrapper controller, please refer to the HSPI documentation.

Table 93 SDIO AHB bus registers

offset address	name	Bit wide	access	description	reset value
0X0000			RO	Rsv	
0X0004					
0X0008	CIS function0 address	[31:0]	RW	The offset address of CIS0 stored in the internal memory of the system. CIS0 actual storage start address = 0x01000 (read command start address) + the offset address	32'b0
0X000C	CIS function1 address	[31:0]	RW	The offset address of CIS1 stored in the internal memory of the system. CIS1 actual storage start address = 0x02000 (read command start address) + the offset address	32'b0
0X0010	CSA address	[31:0]	RW	The cheap address for accessing CSA is set when the RW firmware is initialized. Its principle is the same as CIS settings are the same. CSA is not supported in this design.	32'b0
0X0014	Read address [31:0]	RW		RW is used to set the starting address for DMA to read data from memory. with 32'b0	

				Combining the Data Port register can realize the internal memory of the system. Read operation (that is, the access address is 0x00+ Read address). in this In the design, this method is not used, so it defaults to 0	
0X0018	Write address [31:0] RW is used to set the starting address of DMA writing data to memory. Cooperate			The Data Port register can write to the internal memory of the system operation (that is, the access address is 0x00+ write address). in this setting In the calculation, this method is not used, so the default is 0	32'b0
0X001C	AHB Transfer count	[20:0] RW		The SDIO device notifies the host that in the read data operation to be initiated, it needs to How many bytes of data to read. [23:21] RFU	32'b0
0X001F		RO	--	rsv	
0X0020	SDIO Transfer count	[20:0] RO	The bytes sent from the host to the SDIO device during a data transfer number. When the data transfer is completed, the internal high-speed device through this register The number of bytes sent. [31:21] RFU	32'b0	
0X0024	CIA register	--	RW	[3:0]: CCCR Revision. Default is 4'h2 4'h0 CCCR/FBR Version 1.00 4'h1 CCCR/FBR Version 1.10 4'h2 CCCR/FBR Version 1.20 [7:4] SDIO Revision. Default is 4'h3 4'h0 SDIO Specification 1.00	32'h06017 232

				<p>4'h1 SDIO Version 1.10</p> <p>4'h2 SDIO Version 1.20</p> <p>4'h3 SDIO Version 2.0</p> <p>[11:8] SD Revision. Default is 4'h2</p> <p>4'h0 SD Physical Specification 1.01</p> <p>4'h1 SD Physical-Spec-1.10</p> <p>4'h2 SD Physical Spec 2.0</p> <p>[15:12] IO-Device Code. The default is 4'h7, that is, this product is Wi-Fi device.</p> <p>[16]csa_support, the default is 1.</p> <p>0: CSA is not supported</p> <p>1: Support CSA</p> <p>During initialization, firmware needs to configure this register to 0.</p> <p>[24:17],Extended IO-device code</p> <p>The default is 8'b0.</p> <p>An extension of the standard IO-device code for Fn1.</p> <p>[25], SPS, the default is 1, that is, Fn1 supports high power consumption</p> <p>0: not supported</p>	
--	--	--	--	---	--

				<p>1: Support</p> <p>[26], SHS, default is 1, support high speed</p> <p>0: not supported</p> <p>1: Support</p> <p>[31:27]: RFU</p>	
0X0028	Program Register	--	RW	<p>[0], function ready. hc8051 is required to complete SD initialization</p> <p>After the desired Fn1 register is assigned, set this register to send the SD host to the</p> <p>Indicates that Fn1 is ready to work. Default is 0</p> <p>0: Fn1 not ready</p> <p>1: Fn1 ready</p> <p>[1], fun1 read data ready. Fn1 is ready to send to SD host</p> <p>When sending data, set this register. After the host responds to the read interrupt, the read</p> <p>Interrupt source register (Interrupt Identification), this bit from</p> <p>Move to zero. Default is 0.</p> <p>0: No data is sent to the host</p> <p>1: There is data sent to the host.</p> <p>[2], SCSI. Continuous SPI interrupts are supported. The default is 1.</p> <p>0: not supported</p> <p>1: Support</p> <p>[3], SDC. Indicates that the SDIO card supports execution at the same time as data transmission</p> <p>CMD52 command. Default is 1</p> <p>0: not supported</p>	16'h 02fc

					<p>1: Support</p> <p>[4], SMB. SDIO card supports CMD53 block transmission. default is 1.</p> <p>0: not supported</p> <p>1: Support</p> <p>[5], SRW. Indicates that the SDIO card supports read wait. The default is 1.</p> <p>0: not supported</p> <p>1: Support</p> <p>[6], SBS. Indicates that the SDIO card supports suspend/resume. Default is 1</p> <p>0: not supported</p> <p>1: Support</p> <p>[7], S4MI. Indicates that the SDIO card supports multi-block data transmission in 4bit</p> <p>An interrupt is generated on input. Defaults to 1.</p> <p>0: not supported</p> <p>1: Support</p> <p>[8], LSC. Indicates that the SDIO card is a low-speed device. Default is 0.</p> <p>0: Full speed device</p> <p>1: low-speed equipment</p> <p>[9], 4BLS. Indicates that the SDIO card is a low-speed device, but supports 4bit data transmission. Defaults to 1.</p> <p>0: not supported</p> <p>1: Support</p>	
--	--	--	--	--	--	--

				[10], card ready. Signals belonging to the SD clock domain, power-on reset After release, this register automatically becomes 1, indicating SDIO (interface part points) is ready. When the firmware detects this signal, it can configure the initial Fn1 register required for initialization. 0 at power-on reset. [15:11], RFU. Default is 0	
0X0034	OCR register	--	RW [23:0], working condition register, internally programmable, mainly used to communicate with The host operating voltage range is matched. Default is: 24'hff8000.	Register Bit Supported Voltage Range 0-3 Reserved 4 Reserved 5 Reserved 6 Reserved 7 Reserved 8 2.0-2.1 9 2.1-2.2 10 2.2-2.3 11 2.3-2.4 12 2.4-2.5 13 2.5-2.6 14 2.6-2.7 15 2.7-2.8	32'h00ff8000

				16 2.8-2.9 17 2.9-3.0 18 3.0-3.1 19 3.1-3.2 20 3.2-3.3 3.3-3.4 3.4-3.5 3.5-3.6 [31:24], 8'b0	
0X0038		RW	--	rsv	32'h0
0X003C	CD_State Register	--	RW	[0], indicating the state of the pull-up resistor on the SD dat[3] data line. 0: Pull-up is valid 1: Pull-up is invalid The on-chip AHB bus can access this register. Note: AHB will indirectly modify the result of the write operation to this register CCCR7[7] Value of CD. [31:1], RFU	32'b0
0X0040	Fn1_Ena Register	--	RW	[0], indicates whether Fn1 is enabled. 0: Disable 1: enable	32'b0

				<p>The on-chip AHB bus can access this register.</p> <p>Note: AHB will indirectly modify the result of the write operation to this register</p> <p>CCCR1[1] Value of IOE1.</p> <p>[31:1], RFU</p>	
--	--	--	--	---	--

12 HSPI/SDIO Wrapper Controller

12.1 Function overview

Cooperate with the interface controller (SDIO and HSPI) to complete the DMA operation of data between the host and the internal cache of the chip. Including upstream and downstream data buffering

software and hardware interaction control, filling and releasing of sending and receiving buffers, generation of uplink data interruption, etc.

Both SDIO and HSPI exchange data with the host computer through the wrapper controller, and their control commands and processes are the same. in order to describe

For convenience, some registers are prefixed with SDIO. The corresponding register operations also apply to HSPI

It should be noted that for the prefix SDIO_TX related registers, the control is to receive data from the device. For SDIO_RX related registers,

The control is that the device sends data.

For the cmd field that appears in the register, it is only distinguished from the data frame in terms of description. It does not mean that the command channel can only transmit commands.

used to transmit data. The difference between command and data here is that the command channel has only one buffer area, and the buffer length is generally less than 256

Bytes, and the data channel has multiple buffers, each of which is more than 1K in length. Multiple buffers form a linked list structure with specific lengths.

Configured by software. Due to this linked list cache structure of the data frame, the transfer rate will be faster than the command channel.

12.2 Main Features

- ÿ Support word-aligned data movement

- ÿ Support DMA function

- ÿ Support linked list structure management

- ÿ Support interrupt generation

- ÿ Receive up to 4096 bytes of data

12.3 Functional Description

12.3.1 Uplink data receiving function

The upstream direction refers to the direction in which the master device (SDIO or HSPI) sends data to the slave device (W800).

When the master device sends data to the slave device, after the SDIO or HSPI module receives the data, it will link the data to the interface through WRAPPER.

BD is received, and an interrupt is generated to notify the application software to process the data.

Receive BD descriptor:

sdio_rxbd			
31	0		
Vld[31]	RSV		word0
Sdio_rx_info_size[31:26]	Frm_len[25:12]	Rdbuf0_offset[11:0]	word1
	Sdio_rxbuf0_addr[31:0]		word2
	Sdio_rxbuf1_addr[31:0]		word3
	Sdio_rxbuf2_addr[31:0]		word4
	Next_sdio_rxbd_addr[31:0]		word5

Description: 1.vld, 1 is valid, indicating that the current descriptor points to a valid received frame. 2.rdbuf0_offset: byte address, word alignment, indicating the offset address of the word where the first valid byte of the 802.3 frame is located relative to sdio_rxbuf0. 3. sdio_rxbuf0_addr: byte address, word alignment, buf base address of the first fragment of the upstream data frame. 4. sdio_rxbuf1_addr: byte address, word alignment, buf base address of the second fragment of the upstream data frame. 5. sdio_rxbuf2_addr: byte address, word alignment, buf base address of the third slice of the upstream data frame. 6. next_sdio_rxbd_addr, byte address, word alignment, base address of the next sdio_rxbd. 7.frm_len, byte length, indicates the length of upstream data, excluding sdio_rx_info.

8. Sdio_rx_info_size, byte length, indicates the number of sdio_rx_info bytes, which must be an integer multiple of 4 bytes. Note: The longest frame received is 4096, occupying up to three fragments. Rdbuf0_offset is only valid for the first slice of the frame (that is, the slice in sdio_rxbuf0), and for the remaining two slices, the data is stored sequentially from the base address. The hardware should determine whether the frame exists in multiple fragments according to the upstream data length and the fixed size of 1600 bytes for each buf.

Figure 26 SDIO receives BD descriptor

When the SDIO module or HSPI module of W800 detects that the receiving enable is valid, it reads RXBD and judges the Vld flag.

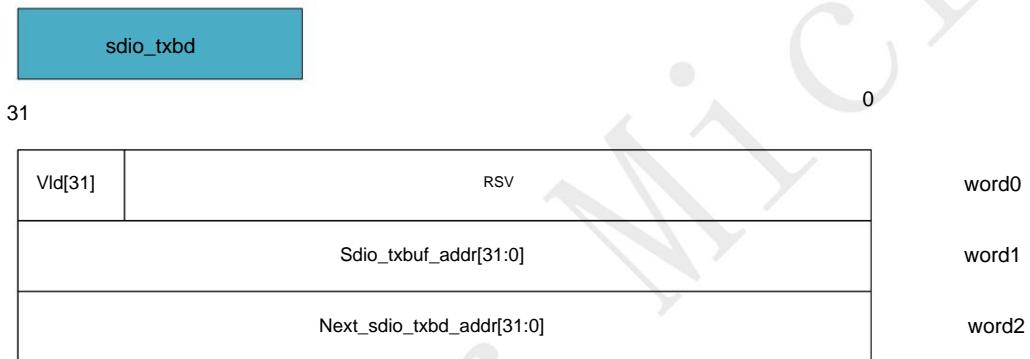
12.3.2 Downlink data transfer function

When the W800 has data to send to the master device, the software first prepares the sending description, and then notifies WRAPPER to send the data

Move, WRAPPER notifies the master device to read the device to be sent through the interrupt signal of SDIO or HSPI, when the data transmission is completed,

WRAPPER generates a send completion interrupt notification routine.

Send BD descriptor:



Description: 1.vld, 1 is valid, indicating that the current descriptor points to the available send buffer.

2. Sdio_txbuf_addr, byte address, must be word aligned. Indicates that this descriptor points to the base address where the transmitted data is stored. This address is offset relative to the base address of each sdio_txbuf to ensure that the base address of the sdio_txbuf is not exceeded when the firmware adds LLC upwards at the beginning of the frame. 3. next_sdio_txbd_addr, byte address, word alignment. The base address of the next sdio_txbd.

Figure 27 SDIO send BD descriptor

12.4 Register Description

12.4.1 Register List

Table 94 WRAPPER Controller Registers

offset address	name	abbreviation	access	describe	reset value
0X0000	WRAPPER interrupt status register INT_STTS		RW command or data frame interrupt status		0X0000

0X0004	WRAPPER interrupt configuration register INT_MASK		Whether the RW command or data frame interrupt is masked	0X0000
0X0008	WRAPPER Uplink command ready to send register	UP_CMD_AVAIL	RW Whether the upstream command is ready	0X0000
0X000c	WRAPPER downlink command buf thread register	DOWN_CMD_BUF_A VAIL	RW Whether the downlink command buf is ready	0X0000
0X0010	SDIO_TX Link Enable Register	SDIO_TX_BD_LINK_E N	RW indicates whether the sdio_txbd linked list descriptor is efficient	0X0001
0X0014	SDIO_TX Link Address Register SDIO_TX_BD_ADDR		RW The address currently pointed to by sdio_txbd, which needs to be To be word-aligned, configuration is required during initialization	0X0000
0X0018	SDIO_TX enable register	SDIO_TX_EN	RW SDIO transmit frame enable	0X0000
0X001c	SDIO_TX Status Register	SDIO_TX_STTS	RO SDIO send status	0X0000
0X0020	SDIO_RX Link Enable Register	SDIO_RX_BD_LINK_E N	RW indicates whether the sdio_rxbd linked list descriptor is efficient	0X0001
0X0024	SDIO_RX Link Address Register SDIO_RX_BD_ADDR		RW The address currently pointed to by sdio_rxbd, which needs to be To be word-aligned, configuration is required during initialization	0X0000
0X0028	SDIO_RX enable register	SDIO_RX_EN	RW SDIO Receive Frame Enable	0X0000
0X002c	SDIO_RX Status Register	SDIO_RX_STTS	RO SDIO receive status	0X0000
0X0030	WRAPPER CMD BUF base address register	CMD_BUF_BASE_AD DR	RW downlink cmd buf base address	0X0000
0X0034	WRAPPER CMD BUF SIZE register	CMD_BUF_SIZE	RW Cmd buf size in bytes	0X0064

12.4.2 WRAPPER INTERRUPT STATUS REGISTER

Table 95 WRAPPER Interrupt Status Register

bit	access	Instructions	reset value
[31:4] RO		reserve	
[3]	RW	int_down_cmd Downlink cmd frame completion interrupt. Write 1 to clear 0.	1'b0
[2]	RW	int_up_cmd Up cmd frame word completion interrupt. Write 1 to clear 0.	1'b0
[1]	RW	int_sdio_txfrm Downlink data frame completion interrupt. Write 1 to clear 0.	1'b0
[0]	RW	int_sdio_rxfrm Upstream data frame completion interrupt. Write 1 to clear 0.	1'b0

12.4.3 WRAPPER INTERRUPT CONFIGURATION REGISTER

Table 96 WRAPPER Interrupt Configuration Register

bit	access	Instructions	reset value
[31:4] RO		reserve	
[3]	RW	int_mask_down_cmd Downlink cmd frame completion interrupt mask register. 1 is shielding, the same below.	1'b0
[2]	RW	int_mask_up_cmd Up cmd frame completion interrupt mask register.	1'b0
[1]	RW	int_mask_sdio_txfrm Downstream data frame completion interrupt mask register.	1'b0
[0]	RW	int_mask_sdio_rxfrm Upstream data frame completion interrupt mask register.	1'b0

12.4.4 WRAPPER Upstream Command Ready Register

Table 97 WRAPPER Upstream Command Ready Register

bit		access	Instructions	reset value

[31:1]		RO	reserve	
[0]		RW	<p>Firmware sets this bit to 1 when there is an upstream cmd frame. When the upstream cmd transfer is complete, the hardware will automatically clear this bit to 0 and generate an interrupt.</p>	1'b0

12.4.5 WRAPPER downlink command buf ready register

Table 98 WRAPPER downlink command buf ready register

bit	access	Instructions	reset value
[31:1] RO		reserve	
[0]	RW	<p>After sending the downlink cmd, the hardware will clear this bit to 0 and generate an interrupt at the same time. When the firmware has finished processing this command line.</p> <p>After command, set this bit to 1.</p>	1'b0

12.4.6 SDIO TX Link Enable Register

Table 99 SDIO TX Link Enable Register

bit	access	Instructions	reset value
[31:1] RO		reserve	
[0]	RW	<p>sdio_txbd link enable, active high.</p> <p>If this bit is valid, the hardware is processing a sdio_txbd descriptor and directly processing the next descriptor pointed to by next_sdio_txbd_addr. If this bit is invalid, the hardware is processing a descriptor.</p> <p>After sdio_txbd, it will stop immediately.</p> <p>The same applies to HSPI.</p>	1'b1

12.4.7 SDIO TX Link Address Register

Table 100 SDIO TX Link Address Register

bit	access	Instructions	reset value
[31: 0] RW		<p>The current sdio_txbd byte address, the software needs to strictly ensure word alignment, the same below.</p> <p>Initially, the firmware needs to configure this register. After the hardware completes a transmission buf each time, the sdio_txbd</p> <p>The next_sdio_txbd_addr in the descriptor is updated to this register.</p> <p>The same applies to HSPI.</p>	32'h0

12.4.8 SDIO TX Enable Register

Table 101 SDIO TX enable register

bit	access	Instructions	reset value
[31:1] RO		reserve	
[0]	RW	<p>SDIO transmit frame enable, active high.</p> <p>The firmware sets this bit to 1 after completing each descriptor sdio_txbd descriptor to notify the SDIO module</p> <p>A new transmit descriptor exists. When the SDIO module detects that this bit is valid, it starts to read the current</p> <p>sdio_txbd, and complete the sending frame process.</p> <p>Hardware automatically completes the clearing of this register.</p> <p>The same applies to HSPI.</p>	1'b0

12.4.9 SDIO TX Status Register

Table 102 SDIO TX Status Register

bit	access	Instructions	reset value

[31:1] RO		reserve	
[0]	RW	<p>SDIO send status.</p> <p>0: SDIO has stopped the transmit process because there are no transmit descriptors available</p> <p>1: SDIO is in the process of sending</p> <p>The same applies to HSPI.</p>	1'b0

12.4.10 SDIO RX Link Enable Register

Table 103 SDIO RX Link Enable Register

bit	access	Instructions	reset value
[31:1] RO		reserve	
[0]	RW	<p>sdio_rxbd link enable, active high.</p> <p>If this bit is valid, the hardware is processing a sdio_rxbd descriptor and directly processing</p> <p>The next descriptor pointed to by next_sdio_rxbd_addr. If this bit is invalid, the hardware is processing a</p> <p>After sdio_rxbd, it will stop immediately.</p> <p>The same applies to HSPI.</p>	1'b1

12.4.11 SDIO RX Link Address Register

Table 104 SDIO RX Link Address Register

bit	access	Instructions	reset value
[31: 0] RW		<p>Current sdio_rxbd byte address.</p> <p>Initially, the firmware needs to configure this register. After the hardware completes a send buf each time, the sdio_rxbd</p> <p>The next_sdio_rxbd_addr in the descriptor is updated to this register.</p>	32'h0

		The same applies to HSPI.	
--	--	---------------------------	--

12.4.12 SDIO RX Enable Register

Table 105 SDIO RX enable register

bit	access	Instructions	reset value
[31:1] RO		reserve	
[0]	RW	<p>SDIO receive frame enable, active high.</p> <p>The firmware sets this bit to 1 after completing each descriptor sdio_rxbd descriptor to notify the SDIO module</p> <p>A new receive descriptor exists. When the SDIO module detects that this bit is valid, it starts to read the current sdio_txbd, and complete the receiving frame process.</p> <p>Hardware automatically completes the clearing of this register.</p> <p>The same applies to HSPI.</p>	1'b0

12.4.13 SDIO RX Status Register

Table 106 SDIO RX Status Register

bit	access	Instructions	reset value
[31:1] RO		reserve	
[0]	RW	<p>SDIO receive status.</p> <p>0: SDIO has stopped receiving process because there is no valid upstream descriptor and upstream command</p> <p>1: SDIO is in the process of receiving</p> <p>The same applies to HSPI.</p>	1'b0

12.4.14 WRAPPER CMD BUF BASE ADDRESS REGISTER

Table 107 WRAPPER CMD BUF Base Address Register

bit	access	Instructions	reset value
[31: 0] RW		<p>Downstream cmd buf base address.</p> <p>The base address of the upstream cmd buf is the base address plus cmd_buf_size.</p>	32'h0

12.4.15 WRAPPER CMD BUF SIZE register

Table 108 WRAPPER CMD BUF SIZE register

bit	access	Instructions	reset value
[31:12] RO		reserve	
[11: 0] RW		The size of cmd buf in bytes, which must be an integer multiple of 4 bytes.	12'd64

13 SDIO HOST Device Controller

13.1 Function overview

The SDIO HOST device controller provides a digital interface capable of accessing Secure Digital Input Output (SDIO) and MMC cards.

Ability to access SDIO devices and SD card devices that are compatible with the SDIO 2.0 protocol. The main interfaces are CK, CMD and 4 data lines.

13.2 Main Features

- ÿ Compatible with SD Card Specification 1.0/1.1/2.0(SDHC)
- ÿ Compatible with SDIO memory card specification 1.1.0
- ÿ Compatible with MMC specification 2.0~4.2
- ÿ Configurable interface clock rate, support host rate 0~50MHz,
- ÿ Support standard MMC interface
- ÿ Support blocks up to 1024 bytes
- ÿ Support soft reset function
- ÿ Automatic Command/Response CRC generation/check;
- ÿ Automatic data CRC generation/check;
- ÿ Configurable timeout detection;
- ÿ Supports SPI, 1-bit SD and 4-bit SD modes
- ÿ Support DMA data transfer

13.3 Functional Description

13.4 Register Description

13.4.1 Register List

offset site	register name		Description of the name bit width attribute		reset value
0x00	mmc_ctrl	RSV	[15:11] RO		
			[10]	RW SDIO read wait enable '1' : enable SDIO read wait '0' : disable SDIO read wait	1'b0
			[9]	RW SDIO interrupt enable '1' : SDIO interrupt enable '0' : SDIO interrupt disabled	1'b0
			[8]	RW SDIO mode enable '1' : SDIO '0' : SD/MMC	1'b0
			[7]	RW SD/MMC/SDIO interface data width '1' : 4 bits '0' : 1 bit	1'b0
			[6]	RW SD/MMC/SDIO transfer mode '1' : High-Speed Mode '0' : Low-Speed Mode	1'b1
			[5:3]	RW SDIO/MMC/SDIO port clock rate selection Refer to Table 2	3'b000

		[2]	RW	SDIO/MMC/SDIO interface drive mode selection '1' : open_DrainMode '0' :Push-Pull Mode	1'b1
		[1]	RW signal mode selection '1' : Automatically select transfer mode '0' : select using mmc_port register		1'b0
		[0]	RW port mode selection '1' : MMC mode '0' :SPI mode		1'b1
0x04	mmc_io	RSV	RO	[15:10]	6'd0
		RW	[9]	SDIO cmd12/IO Abort flag '1' : mark the current command as cmd12/IO Abort '0' : mark the current command is not cmd12/IO Abort	1'b0
		RW	[8]	SDIO Command Properties '1' : mark the data block after the current command; '0' : mark no data block and command response after the current command answer;	1'b0
		RW	[7]	Enable auto generate 8 null clock after response/command or single block data	1'b0

				Automatically generate 8 after a response/command or a single block of data a null clock function '1' : enable '0' : off	
	RW	[6]	Enable auto receive response after command Automatically receive response function after command '1' : enable '0' : off	1'b0	
	RW	[5]	When there are 8 nulls on the SD/MMC/SDIO port clock line clock generation '1' : generate 8 null clocks '0' : Receive response/transmit command according to bit 3 setting	1'b0	
	RW	[4]	Designed for CID and CSD reading. When sending CID or CSD command, SD/MMC/SDIO card device will be in CMD online reply 136bit CID or 11 CSD data. When this bit is set to 1, CID or CSD The data will be stored at [135:8] in the command buffer area middle;	1'b0	
	RW	[3]	Response/command selection when bit[5] is '0' '1': receive response	1'b0	

					'0': Send command.	
		RW	[2]		<p>Set automatic 8 null clock/command/response transmission</p> <p>Function</p> <p>'1': Enable automatic 8 null clock/command/response transmission</p> <p>'0': Disable automatic 8 null clock/command/response transmission</p> <p>lose.</p> <p>Generate 8 nulls according to bit 5 and bit3 settings</p> <p>Clock, receive response or transmit command, when the biography</p> <p>After the input is completed, this bit is automatically cleared;</p>	1'b0
		RW	[1]		<p>Set data transfer direction</p> <p>'1' : read data; .</p> <p>'0' : write data;</p>	1'b0
		RW	[0]		<p>Set up automatic data transfer</p> <p>'1' : enable automatic data transfer</p> <p>'0' : Disable automatic data transfer.</p> <p>When the data transmission is completed, this bit will be automatically cleared;</p>	1'b0
0x08	mmc_bytecntl		RW	[15:0]	Data transfer byte count register	16'h0200
0x0C	mmc_tr_blockcnt		RO	[15:0]	Completed block counter 16'h0000 when multi-block transfer	
0x10	mmc_crcctl		RW	[7]	<p>SD/MMC/SDIO port CMD Line CRC</p> <p>circuit enable.</p> <p>SD/MMC/SDIO port CMD line CRC function</p>	1'b0

					can '1': enable. '0': off.	
	RW	[6]			SD/MMC/SDIO port data line CRC function '1': enable. '0': off.	1'b0
	RW	[5]			[5] Enable automatic CRC check crc_status '1': enable, when crc_status !=3'b010, A crc status interrupt will be generated, and the write data transfer will be The stop command is interrupted and mmc_io[0] or mmc_io_mbctl[2:0] will be cleared; 0: close;	1'b0
	RW	[4]			Read multi-block function before response '1': enable. '0' : off	1'b0
	RW	[3:2]			DAT CRC selection. DAT CRC selection Refer to Table 4	1'b0
	RO	[1]			CMD CRC error.	1'b0
	RO	[0]			DAT CRC error	1'b0
0x14	cmd_crc	RSV	RO	[7]	RSV	1'b0
			RO	[6:0]	CMD CRC register value	7'd0

0x18	dat_crcl		RO	[7:0]	The DAT CRC low register value	NA
0x1C	dat_crch		RO	[7:0]	The DAT CRC high register value	NA
0x20	mm_port		RW	[7]	SD/MMC/SDIO port Clock line signal.	1'b0
			RW	[6]	SD/MMC/SDIO port CMD line signal	1'b1
			RW	[5]	SD/MMC/SDIO port data line signal	1'b1
			RW	[4]	Automatically check Ncr timeout function '1': Enable automatic check Ncr timeout. '0': Disable automatic check Ncr timeout	1'b1
			RW	[3:0]	Ncr timeout count value (SD/MMC/SDIO clock count).	4'hF
0x24	mmc_int_mask RSV		RO	[15:9]		7'd0
				[8]	SDIO data line 1 interrupt mask '1': do not block '0': mask	1'b0
				[7]	CRC status token error interrupt mask '1': do not block '0': mask	1'b0
				[6]	Command and response Ncr timeout interrupt mask '1': do not block '0': mask	1'b0
				[5]	Multi-block timeout interrupt mask. '1': do not block	1'b0

					'0': mask	
		[4]			Multi-block transfer complete interrupt mask. '1': do not block '0': mask	1'b0
		[3]	Command	CRC error interrupt mask '1': do not block '0': mask		1'b0
		[2]		Data CRC Error Interrupt Mask '1': do not block '0': mask		1'b0
		[1]		Data transfer complete interrupt mask '1': do not block '0': mask		1'b0
		[0]	Command	transfer complete interrupt mask '1': do not block '0': mask		1'b0
0x28	clr_mmc_int	RSV	RO	[15:9]		7'd0
			RW	[8]	W: Clear SDIO data line 1 interrupt R: SDIO data line 1 interrupt status	1'b0
			RW	[7]	W: Clear CRC status token error interrupt R: CRC status token error interrupt status;	1'b0

					When this bit is 1, judge mmc_sig[6:4]	
	RW	[6]	[6] W: Clear command and respond to Ncr timeout interrupt; R: Command and response Ncr timeout interrupt status;		1'b0	
	RW	[5]	[5] W: clear multi-block timeout interrupt; . R: Multi-data block timeout interrupt status;		1'b0	
	RW	[4]	[4] W: Clear multi-block transfer completion interrupt; . R: Multi-data block transfer completion interrupt status;		1'b0	
	RW	[3]	[3] W: Clear command CRC error interrupt; . R: Command CRC error interrupt status;		1'b0	
	RW	[2]	[2] W: clear data CRC error interrupt; . R: Data CRC error interrupt status;		1'b0	
	RW	[1]	[1] W: Clear data transfer completion interrupt; . R: Data transmission complete interrupt status;		1'b0	
	RW	[0]	[0] W: Clear command transfer completion interrupt; . R: Command transmission complete interrupt status;		1'b0	
0x2C	mmc_cardsel	RW	[7]	SD/MMC/SDIO controller enable '1': enable	1'b0	

					'0': off	
		RW	[6]		Enable SD/MMC/SDIO card device clock line '1': enable '0': off	1'b1
		RW	[5:0]		SD/MMC/SDIO time base factor Use this register to build a 1MHz clock; $1\text{MHz} = \text{Fhclk} / ((\text{mmc_cardsel}[5:0] + 1) * 2)$	6'd0
0x30	mmc_sig	RW	[7]		SD/MMC/SDIO port CMD line signal When reading this register, the SDIO controller will A clock pulse is generated on the clock line. and on the clock The state of the CMD line on rising edge will be stored to this register in the device.	1'b1
		RW	[6:4]		CRC status[2:0] When write data CRC status token time;	3'b111
		RW	[3]		SD/MMC/SDIO port DAT3 data signal. 1'b1	
		RW	[2]		SD/MMC/SDIO port DAT2 data signal.	1'b1
		RW	[1]		SD/MMC/SDIO port DAT1 data signal.	1'b1
		RW	[0]		SD/MMC/SDIO port DAT0 data signal.	1'b1
0x34	mmc_io_mbctl	RW	[7:6]		SD/MMC/SDIO NAC timeout range selection 2'b0	

				Refer to Appendix 2 -Table 5.	
	RW	[5:4]		SD/MMC/SDIO Busy timeout scale selection. SD/MMC/SDIO device busy timeout range selection. Refer to Appendix 2 -Table 6	2'd1
	RW	[3]		SD/MMC/SDIO port clock line polarity 1: Send on the falling edge of the clock, and collect on the rising edge; 0: Send on the rising edge of the clock, and collect on the falling edge;	1'b0
	RW	[2]		Set up SD/MMC/SDIO port fully automatic commands and Multi-block transfer '1': enable '0': off Setting this bit to 1 (mmc_io[7:6]==11) will trigger Send an SD/MMC/SDIO command, respond, 8 null clock, multi-block transfer. When the data transfer is complete After completion, this bit will be automatically cleared.	1'b0
	RW	[1]		Select multiple block data transfer direction. Set multi-block transfer direction '1' : read data. '0' : write data.	1'b0

			RW	[0]	<p>Set SD/MMC/SDIO port auto multiple block data transfer.</p> <p>Set SD/MMC/SDIO port automatic multi-blocking transmission</p> <p>'1' : enable.</p> <p>'0' : off.</p> <p>Setting this bit to 1 (mmc_io[7:6]==11) will trigger Send a multi-block transfer of SD/MMC/SDIO.</p> <p>The number of data blocks is in mmc_blocknt set in the register. When the data transmission is completed, this bit will automatically clear.</p>	1'b0
0x38	mmc_blockcnt		RW	[15:0]	<p>Data block number register.</p> <p>data block number register</p> <p>Configuring this register defines a total of</p> <p>The number of data blocks to be transmitted.</p>	16'h0001
0x3C	mmc_timeoutcnt		RW	[7:0]	<p>Data transfer timeout count register.</p> <p>Data transfer timeout counter</p> <p>Time = Scale* bit[7:0].</p> <p>Scale by register</p> <p>mmc_io_mbctl[7:6]/[5:4] definition;</p>	8'h40
0x40	cmd_buf0		RW	[7:0]	<p>The cmd_buf byte 0. Mapped to command line bit [15:8]</p>	8'h00

					Command buf byte 0 , map to the command line bit[15:8]	
0x44	cmd_buf1		RW	[7:0]	The cmd_buf byte 1. Mapped to command line bit [23:16] Command buf byte 1 , map to the command line bit[23:16]	8'h00
0x48	cmd_buf2		RW	[7:0]	The cmd_buf byte 2. Mapped to command line bit [31:24] command buf byte 2 , map to the command line bit[31:24]	8'h00
0x4C	cmd_buf3		RW	[7:0]	The cmd_buf byte 3. Mapped to command line bit [39:32] command buf byte 3 , map to the command line bit[39:32]	8'h00
0x50	cmd_buf4		RW	[7:0]	The cmd_buf byte 4. Mapped to command line bit [47:40] command buf byte 4 , map to the command line bit[47:40]	8'h00
0x54	cmd_buf5		RW	[7:0]	The cmd_buf byte 5. Mapped to command line bit [55:48] Command buf byte 5 , map to the command line bit[55:48]	8'h00

0x58	cmd_buf6		RW	[7:0]	The cmd_buf byte 6. Mapped to command line bit [63:56] command buf byte 6 , map to the command line bit[63:56]	8'h00
0x5C	cmd_buf7		RW	[7:0]	The cmd_buf byte 7. Mapped to command line bit [71:64] command buf byte 7 , map to the command line bit[71:64]	8'h00
0x60	cmd_buf8		RW	[7:0]	The cmd_buf byte 8. Mapped to command line bit [79:72] Command buf byte 8 , map to the command line bit[79:72]	8'h00
0x64	cmd_buf9		RW	[7:0]	The cmd_buf byte 9. Mapped to command line bit [87:80] command buf byte 9 , map to the command line bit[87:80]	8'h00
0x68	cmd_buf10		RW	[7:0]	The cmd_buf byte 10. Mapped to command line bit [95:88] Command buf byte 10, mapped to the command line bit[95:88]	8'h00
0x6C	cmd_buf11		RW	[7:0]	The cmd_buf byte 11. Mapped to command line bit [103:96]	8'h00

					Command buf byte 11, mapped to the command line bit[103:96]	
0x70	cmd_buf12		RW	[7:0]	The cmd_buf byte 12. Mapped to command line bit [111:104] Command buf byte 12, mapped to the command line bit[111:104]	8'h00
0x74	cmd_buf13		RW	[7:0]	The cmd_buf byte 13. Mapped to command line bit [119:112] Command buf byte 13, mapped to the command line bit[119:112]	8'h00
0x78	cmd_buf14		RW	[7:0]	The cmd_buf byte 14. Mapped to command line bit [127:120] Command buf byte 14, mapped to the command line bit[127:120]	8'h00
0x7C	cmd_buf15		RW	[7:0]	The cmd_buf byte 15. Mapped to command line bit [135:128] Command buf byte 15, mapped to the command line bit[135:128]	8'h00
0x80	buf_ctrl		RW	[15]	Data buffer clear enable 1: Trigger data buffer clearing; 0: keep When written to 1, this register is automatically cleared after one clock	1'b0

				zero;	
	RW	[14]		DMA request mask 0: not masked; 1: shield; Note: Please configure this register before enabling dma; Configuring this register after enabling dma has no effect;	1'b0
	RW	[13]	RSV		1'b0
	RW	[12]	Data FIFO Status Signal Mask Configuration 1: Activate 0: default Data FIFO status signal, active high; Mask the FIFO full signal when reading the card device; Shield FIFO empty signal when writing card device;		1'b0
	RW	[11]	Set the buffer access direction 1: write 0: read		1'b0
	RW	[10]	DMA hardware interface enable: 1: Enable DMA interface; 0: AHB interface accesses data cache; When using the DMA interface, when the data transfer is completed, the Automatically reset this bit (single data block transfer or multi-data		1'b0

					block transfer)	
		RW	[9:2] Data cache data watermark settings; only when Valid when buf_ctl[10]=1; Note: The data cache depth is 128 words, not To configure this register greater than 127.		8'd0	
		RO	[1] Data buffer empty signal			1'b1
		RO	[0] Data buffer full signal			1'b0
0x100~ 0x2FF	data_buf	RW		data cache		NA

Bit5	Bit4	Bit3	Speed
0	0	0	1/2 base clock
0	0	1	1/4 base clock
0	1	0	1/6 base clock
0	1	1	1/8 base clock
1	0	0	1/10 base clock
1	0	1	1/12 base clock
1	1	0	1/14 base clock
1	1	1	1/16 base clock

Table 2 Mmc_ctrl[5:3] detailed definition

Note: When mmc_ctrl[6] = 1, when the controller works in high-speed mode, base clk = hclk;

When mmc_ctrl[6] = 0, when the controller works in low speed mode, base clk = clk1m;

Clk1m= Fhclk/((mmc_cardsel[5:0]+1)*2);

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Operation mode	transfer byte	
x	x	x	x	x	0	x	0	no action		N/A
x	x	1	x	0	Trig x		0	generates 8 null clk		N/A
0	0	0	x	0	Trig x		0	send command		6
0	0	0	0	1	Trig x		0	Receive response		6
0	0	0	1	1	Trig x		0	Receive response		17
1	0	0	0	0	Trig x		0	transmit command + generate 8 null clk		N/A

0	1	0	0	0	Trig x		0 transmit command + receive response answer	N/A
1	1	0	0	0	Trig x		0 transmit command + receive response should + generate 8 null clk	N/A
x	x	x	x	x	0	1	Trig reads a single data +8 null clk	mmc_bytectn
x	x	x	x	x	0	0	Trig write single data +8 null clk	mmc_bytectn

Table 3 Mmc_io[7:0] detailed definition

Note:

1. Except for the last two lines in Table 3, other operations will generate CMD DONE interrupt;
2. The last two lines of operations in Table 3 will generate a data transfer completion interrupt.

Bit3	Bit2	data_crcl and data_crch registers display content
0	0	DAT0 line data CRC value
0	1	DAT1 line data CRC value
1	0	DAT2 line data CRC value
1	1	DAT3 line data CRC value

Table 4 mmc_crctrl[3:2] detailed definition

Bit7	Bit6	Bit2	Bit1	Bit0	Operation Description	transfer bytes
mmc_io		mmc_io_mbctl				
1	1	Trig	0	0 write	multi-block command + response + 8 null clock + data	mmc_blockcnt
1	1	Trig	1	0 read	multi-block command + response + 8 null clock + data	mmc_blockcnt
x	x	0	0	Trig	Write multiple blocks of data	mmc_blockcnt
x	x	0	1	Trig	Read multiple pieces of data	mmc_blockcnt

Table 5 mmc_io[7:6] and mmc_io_mbctl[2:0] detailed definition

Note:

1. The operations in the first two columns in Table 5 will generate multiple data completion interrupts, each block of data will generate a data completion interrupt, and

CMD DONE interrupt;

2. When a timeout occurs, the multi-block data completion interrupt will not be generated, but a timeout interrupt will be generated.

Bit7	Bit6	time unit
0	0	1us
0	1	100us
1	0	10ms
1	1	1s

Table 6 mmc_io_mbctl[7:6] NAC timeout interrupt unit selection

Bit7	Bit6	time unit
0	0	1us
0	1	100us
1	0	10ms
1	1	1s

Table 7 mmc_io_mbctl[5:4] Port timeout interrupt unit selection

Note:

1. When using the DMA interface, the DMA enable needs to be turned on first;

2. If interrupts are not used, mmc_io[2] can be queried when transmitting command/response/8 null clock;

When transferring data, you can query mmc_io[0]; when transferring multiple pieces of data, you can query mmc_io_mbctl[2]

/mmc_io_mbctl0];

3. When the Ncr timeout occurs, the data transmission will be interrupted, and the controller needs to reconfigure a new transmission;

Winner Micro

14 SPI Controller

14.1 Function overview

SPI is an abbreviation for Serial Peripheral Interface. SPI is a high-speed, full-duplex, synchronous communication protocol

Wire. The communication principle of SPI is very simple. It works in a master-slave mode. This mode usually has a master device and one or more slave devices.

At least 4 wires, in fact 3 wires are also possible (for unidirectional transmission), including SDI (data input), SDO (data output), SCLK (time clock), CS (chip select).

14.2 Main Features

- ÿ Can be used as SPI master or SPI slave
- ÿ Transmit and receive paths each have 8-word deep FIFOs
- ÿ master supports 4 formats of motorola spi (CPOL, CPHA), TI timing, macrowire timing
- ÿ slave supports 4 formats of motorola spi (CPOL, CPHA)
- ÿ Supports full duplex and half duplex
- ÿ The main device supports bit transmission, the maximum supports 65535bit transmission
- ÿ The slave device supports transmission modes of various length bytes
- ÿ The maximum clock frequency of spi_clk input from the device is 1/6 of the system APB clock

14.3 Functional Description

14.3.1 Master and slave can be configured

The SPI controller supports both the device as the SPI communication master and the device as the SPI communication slave. By setting the SPI_CFG register

Bit2 of the device can switch the master-slave role of the device back and forth.

14.3.2 Multiple Mode Support

As a master device, by setting Bit1 (CPHA) and Bit0 (CPOL) of the SPI_CFG register, it can be set to MOTOROLA respectively.

The four formats of SPI transmit data. CPOL is used to determine the idle level of the SCK clock signal, CPOL=0, the idle level is low,

When CPOL=1, the idle level is high. CPHA is used to determine the sampling time, CPHA=0, on the first clock edge of each cycle

Sampling, CPHA=1, is sampled on the second clock edge of each cycle. The master can also be set by setting the TRANS_MODE register

The data is transmitted in TI timing or microwire timing, and the transmission data length under both timings can be adjusted.

As a slave device, only four formats of MOTOROLA SPI are supported, and the format selection is also done by setting the same signal as that of the master device.

register to achieve.

14.3.3 Efficient data transfer

The FIFO memory is a first-in-first-out dual-port buffer, that is, the first data entered into it is the first to be shifted out, and one of the memory's

The input port, and the other port is the output port of the memory. The SPI controller integrates two (one for each transceiver) FIFO storage with a depth of 8 words

It can increase the data transfer rate, handle a large number of data streams, and match systems with different transfer rates, thereby improving system performance. able to pass

Setting Bit[8:6] and Bit[4:2] of the MODE_CFG register can set the trigger level of RXFIFO and TXFIFO to meet different

performance requirements at the same transmission rate. After the trigger level of the FIFO is triggered, an interrupt or DMA can be triggered to transfer the data from the memory

Move to TXFIFO or move data from RXFIFO to memory.

14.4 Register Description

14.4.1 Register List

Table 109 SPI register list

offset address name		abbreviation	access	describe	reset value
0X0000	Channel configuration register CH_CFG		RW	RW is used to perform some configuration	0X0000_0000
0X0004	SPI configuration register SPI_CFG		RW	Configure SPI communication related items	0X0000_0004
0X0008	Clock configuration register Clk_CFG		RW	RW is used to set the clock frequency division factor	0X0000_0000
0X000C	Mode configuration register MODE_CFG		RW	RW configuration transfer mode	0X0000_0000
0X0010	Interrupt Control Register SPI_INT_MASK		RW	RW mask or enable related interrupt	0X0000_00FF
0X0014	Interrupt status register SPI_INT_SOURCE	RW	RW is used to query the interrupt source		0X0000_0000
0X0018	SPI Status Register SPI_STATUS		RO	RO enumerates relevant states in SPI communication	0X0000_0000
0X001C	SPI Timeout Register SPI_TIME_OUT		RW	RW Set SPI communication timeout	0X0000_0000
0X0020	Data transmission register SPI_TX_DATA		RW	RW TX FIFO, used to store the data to be sent	0X0000_0000
0X0024	Transfer mode register TRANS_MODE		RW	RW Set transfer mode	0X0000_0000
0X0028	Data length register SLV_XMIT_LEN		RO	When RO is used as a slave device, it is used to store and send out or The length of the data received by the	0X0000_0000
0X002C		RSV		reserve	0X0000_0000
0X0030	Data receiving register SPI_RX_DATA		RW/RO	RW RX FIFO, used to store the received data	0X0000_0000

14.4.2 Channel Configuration Register

Table 110 SPI Channel Configuration Register

bit access		Instructions	reset value
[31]		RSV	1'b0
[30:23] RW		RX_INVALID_BIT	8'h0

		<p>Indicates how many first bits are invalid data when the receiving channel starts to receive, and these invalid data need to be thrown directly off, do not enter the Rx FIFO. Only subsequent data goes into the Rx FIFO</p> <p>This register is used in conjunction with Tx/Rx length. The final amount of data actually stored in the Rx FIFO is Tx/Rx length - RX_INVALID_BIT</p> <p>Note: master mode is valid</p> <p>Motorola/TI mode active</p>	
[twenty two]	RW	<p>Clear FIFOs, clear the contents of Tx and Rx FIFO, and simultaneously reset all circuits of master and slave (except configuration registers)</p> <p>1'b0: do not clear the FIFO</p> <p>1'b1: Clear valid</p> <p>Set to 1 by software, cleared to 0 by hardware</p> <p>Note: Both master/slave are valid</p> <p>Motorola/TI/microwire mode is valid</p>	1'b0
[twenty one]	RW	<p>continue mode, in this mode, the spi transmission is not affected by the empty Tx FIFO. , continue to transmit until the whole The transfer process is complete.</p> <p>1'b0: normal, after the Tx FIFO is empty, you need to wait for data to appear in the FIFO, and the SCK stops flipping. Similarly, After the Rx FIFO is full, SCK stops flipping and waits for the RX FIFO to have space to receive data</p> <p>1'b1: continue mode, if the Tx fifo is empty, it can still transmit until the transmission is completed, but if the rx fifo If it is full, you need to suspend the transmission until the rx fifo can store the number</p> <p>Note: master is valid</p> <p>Normally, this mode is not set.</p>	1'b0

		<p>When this mode is enabled, if there is no data in the tx fifo, invalid data may be sent first. so please</p> <p>After filling in the data, start the spi master</p> <p>Motorola/TI/microwire mode is valid</p>	
[20]	RW	<p>RxChOn, whether the receive channel is on</p> <p>1'b0: Rx channel off</p> <p>1'b1: Rx channel on</p> <p>Note: Both master/slave are valid</p> <p>Motorola/TI/microwire mode is valid</p>	1'b0
[19]	RW	<p>TxChOn, whether the transmission channel is turned on</p> <p>1'b0: Tx channel off</p> <p>1'b1: Tx channel on</p> <p>Note: Both master/slave are valid</p> <p>Motorola/TI/microwire mode is valid</p>	1'b0
[18:3]	RW	<p>Tx/Rx length</p> <p>When Spi is transmitting, the number of valid SCKs</p> <p>It also indirectly reflects the length of the data sent or received.</p> <p>When (TxChOn=1, RxChOn=1), it indicates the number of bits sent and the maximum number of bits received (with How much the body receives is related to RX_INVALID_BIT)</p> <p>When (TxChOn=1, RxChOn=0),</p> <p>Indicates the number of bits sent,</p> <p>When (TxChOn=0, RxChOn=1),</p> <p>Indicates the maximum number of bits received (the specific number of received bits is related to RX_INVALID_BIT, the actual number of received bits is Tx/Rx</p>	16'h0

		<p>length - RX_INVALID_BIT)</p> <p>When (TxChOn=0, RxChOn=0),</p> <p>meaningless.</p> <p>Note: master is valid</p> <p>Motorola/TI/microwire mode is valid</p>	
[2]	RW	<p>Chip selects</p> <p>1'b0: SPI_CS valid signal is 0</p> <p>1'b1: SPI_CS valid signal is 1</p> <p>Note: master is valid</p> <p>Motorola/TI/microwire mode is valid</p>	1'b0
[1]	RW	<p>Force CS out</p> <p>1'b0: spi_cs signal output is controlled by hardware</p> <p>1'b1: The spi_cs signal output is controlled by software, and the specific output value is Chip selects</p> <p>This signal cooperates with Chip selects to realize the programmable output csn signal, that is, when the signal is 1, spi_cs = Chip selects</p> <p>Note: master is valid</p> <p>Motorola/TI/microwire mode is valid</p>	1'b0
[0]	RW	<p>SPI start,</p> <p>Command SPI to start receiving or sending, write 1 for spi to start working, after that, it will automatically return to zero</p> <p>1'b0: stop spi working</p> <p>1'b1: Start a send or receive of spi, automatically return to zero</p> <p>Note: master is valid</p>	1'b0

		Motorola/TI/microwire mode is valid	
--	--	-------------------------------------	--

14.4.3 SPI Configuration Register

Table 111 SPI configuration registers

bit access		Instructions	reset value
[31:19]		RSV	13'h0
[18:17] RW		FRAM FORMAT 2'b00: motorola 2'b01:TI 2'b10: microwire 2'b11: RSV Select the master to support the protocol of that manufacturer Note: master is valid	2'b0
[16]	RW	SPI_TX pin always driven 1'b0: spi output is driven only when spi_cs is valid, and is tri-stated at other times 1'b1: The spi output is always driven, even if there is no data transfer Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1'b0
[15]		RSV	1'b0
[14:12] RW		cs hold, the time that spi_cs is valid after data transmission is completed, that is, the hold time of spi_cs 3'b000 >=1 APB bus CLK 3'b001 >=2 APB bus CLK 3'b010 >=4 APB bus CLK	3'b0

		3'b011 >=8 APB bus CLK 3'b100 >=16 APB bus CLK 3'b101 >=32 APB bus CLK 3'b110 >=64 APB bus CLK 3'b111 >=127 APB bus CLK Note: master is valid Motorola mode is valid	
[11:9] RW		cs setup, the time that spi_cs is valid in advance before data transmission, that is, the setup time of spi_cs 3'b000 >=1 APB bus CLK 3'b001 >=2 APB bus CLK 3'b010 >=4 APB bus CLK 3'b011 >=8 APB bus CLK 3'b100 >=16 APB bus CLK 3'b101 >=32 APB bus CLK 3'b110 >=64 APB bus CLK 3'b111 >=127 APB bus CLK Note: master is valid Motorola mode is valid	3'b0
[8:7] RW		SPI-out delay, the delay of SPI data output relative to SCK, mainly for the consideration of hold time. [8:7] Number of system clock cycles (APB clock) 2'b00 0 2'b01 1	2'b0

		<p>2'b10 2</p> <p>2'b11 3</p> <p>Note: Both master/slave are valid</p> <p>Motorola mode is valid</p>	
[6:4] RW		<p>Frame delay, the default interval between the end of a frame (spi_cs is valid) and the beginning of the next frame is SCK</p> <p>Half of the clock period, ie SPI_CS inactive time. But for compatibility, it can be configured here. Default at least 0.5SCK</p> <p>[6:4] SCK clock</p> <p>3'b000 0</p> <p>3'b001 2</p> <p>3'b010 4</p> <p>3'b011 8</p> <p>3'b100 16</p> <p>3'b101 32</p> <p>3'b110 64</p> <p>3'b111 127</p> <p>For example, if 128byte data is transmitted in block mode, after the data transmission is completed, the set delay will be added.</p> <p>late time.</p> <p>Note: master is valid</p>	3'b0
[3]	RW	<p>Big endian</p> <p>1'b0: The data format adopts the little endian mode, that is, during the transmission process, the low byte is sent first</p> <p>1'b1: The data format adopts the big endian mode, that is, during the transmission process, the high byte is sent first</p>	1'b0
[2]	RW	MASTER/SLAVE	1'b1

		<p>1'b0: slave, the device is slave</p> <p>1'b1: master, the device is the master</p> <p>Note: Both master/slave are valid</p>	
[1]	RW	<p>SPI CPHA</p> <p>1'b0: Transmission Mode A</p> <p>1'b1: transfer mode B</p> <p>Note: Both master/slave are valid</p> <p>Motorola mode is valid</p>	1'b0
[0]	RW	<p>SPI CPOL, SCK polarity at IDLE</p> <p>1'b0: 0 when SCK IDLE</p> <p>1'b1: 1 for SCK IDLE</p> <p>Note: Both master/slave are valid</p> <p>Motorola mode is valid</p>	1'b0

14.4.4 Clock Configuration Register

Table 112 SPI Clock Configuration Register

bit access		Instructions	reset value
[31:16]		RSV	16'h0
[15:0] RW		<p>Divider</p> <p>$FSCK = FAPB_CLK / (2 \times (\text{Divider} + 1))$</p> <p>Note: master is valid</p> <p>Motorola/TI/microwire mode is valid</p>	16'h0
[31:16]		RSV	16'h0

[15:0] RW		<p>Divider</p> <p>$FSCK = FAPB_CLK / (2 \times (\text{Divider} + 1))$</p> <p>Note: master is valid</p> <p>Motorola/TI/microwire mode is valid</p>	16'h0
-----------	--	---	-------

14.4.5 Mode Configuration Register

Table 113 SPI Mode Configuration Register

bit access		Instructions	reset value
[31:9]		RSV	23'h0
[8:6] RW		<p>RxTrigger level</p> <p>Data stored in RX FIFO triggers interrupt or DMA request threshold: 0~7word</p> <p>Only when the data in the rxbuffer is greater than the RxTrigger level will trigger an interrupt or request a DMA transfer</p> <p>Note: Both master/slave are valid</p> <p>Motorola/TI/microwire mode is valid</p>	3'b0
[5]		RSV	1'b0
[4:2] RW		<p>TxTrigger level</p> <p>Data stored in TX FIFO triggers interrupt or DMA request threshold: 0~7word</p> <p>Only when the data in the txbuffer is greater than or equal to the TxTrigger level will trigger an interrupt or request a DMA transfer.</p> <p>shift</p> <p>Note: Both master/slave are valid</p> <p>Motorola/TI/microwire mode is valid</p>	3'b0
[1]	RW	RxDMA On, use DMA to move data enable	1'b0

		<p>1'b0: Do not use DMA, 1'b1: DMA Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	
[0]	RW	<p>TxDMA On, use DMA to move data enable 1'b0: Do not use DMA, 1'b1: DMA Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b0

14.4.6 Interrupt Control Register

Table 114 SPI Interrupt Control Register

bit access		Instructions	reset value
[31:8]		RSV	24'h0
[7]	RW	<p>IntEn_spi_timeout 1'b0: Enable to generate spi_timeout interrupt 1'b1: spi_timeout interrupt is not allowed Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b1
[6]	RW	<p>IntEn_spi_done 1'b0: spi send or receive completed, enable interrupt generation 1'b1: spi send or receive is completed, interrupts are not allowed</p>	1'b1

		Note: Both master/slave are valid Motorola/TI/microwire mode is valid	
[5]	RW	<p>IntEnRxOverrun</p> <p>1'b0: Rx FIFO overflow interrupt enable</p> <p>1'b1: Rx FIFO overflow interrupt disabled</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b1
[4]	RW	<p>IntEnRxUnderrun</p> <p>1'b0: Rx FIFO underflow interrupt disabled</p> <p>1'b1: Rx FIFO underflow interrupt enable</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b1
[3]	RW	<p>IntEnTxOverrun</p> <p>1'b0: Tx FIFO overflow interrupt enable</p> <p>1'b1: Tx FIFO overflow interrupt disabled</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b1
[2]	RW	<p>IntEnTxUnderrun</p> <p>1'b0: Tx FIFO underflow interrupt enable</p> <p>1'b1: Tx FIFO underflow interrupt disabled</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b1
[1]	RW	<p>IntEnRxFifoRdy</p> <p>1'b0: Rx FIFO has data upload interrupt enable</p> <p>1'b1: Rx FIFO has data upload interrupt disabled</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b1
[0]	RW	IntEnTxFifoRdy	1'b1

		<p>1'b0: Tx FIFO can write data to TX FIFO interrupt enable</p> <p>1'b1: Tx FIFO can write data to TX FIFO interrupt disabled</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	
--	--	--	--

14.4.7 Interrupt Status Register

Table 115 SPI Interrupt Status Register

bit access		Instructions	reset value
[31:8]		RSV	24'h0
[7]	RW	<p>spi_timeout</p> <p>1'b0: There is no end data in rxfifo that needs to be taken by the CPU</p> <p>1'b1: There is end data in rxfifo that needs to be taken by the CPU</p> <p>Write 1 to clear</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b0
[6]	RW	<p>spi_done</p> <p>1'b0: SPI transmission or reception is not completed</p> <p>1'b1: SPI send or receive completed</p> <p>Write 1 to clear</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b0
[5]	RW	<p>RxOverrun</p> <p>1'b0: Rx FIFO overflow</p> <p>1'b1: Rx FIFO overflow</p> <p>Write 1 to clear</p>	1'b0

		Note: Both master/slave are valid Motorola/TI/microwire mode is valid	
[4]	RW	<p>RxUnderrun</p> <p>1'b0: Rx FIFO underflow</p> <p>1'b1: Rx FIFO underflow</p> <p>Write 1 to clear</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b0
[3]	RW	<p>TxOverrun</p> <p>1'b0: Tx FIFO overflow</p> <p>1'b1: Tx FIFO overflow</p> <p>Write 1 to clear</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b0
[2]	RW	<p>TxUnderrun</p> <p>1'b0: Tx FIFO underflow</p> <p>1'b1: Tx FIFO underflow</p> <p>Write 1 to clear</p> <p>With continue mode = 1, the interrupt is never generated.</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b0
[1]	RW	<p>RxFifoRdy</p> <p>1'b0: Rx FIFO data volume <= RxTrigger level, no need to upload</p> <p>1'b1: Rx FIFO data volume > RxTrigger level, request to upload</p> <p>Write 1 to clear</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b0

[0]	RW	<p>TxFifoRdy</p> <p>1'b0: Tx FIFO data volume > TxTrigger level, cannot write data to TX FIFO</p> <p>1'b1: Tx FIFO data volume <= TxTrigger level, data can be written to TX FIFO</p> <p>Write 1 to clear</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b0
-----	----	--	------

14.4.8 SPI Status Register

Table 116 SPI Status Register

bit access		Instructions	reset value
[31:13]		RSV	19'h0
[12]	RO	<p>SPI Busy</p> <p>1'b0: SPI has no transmit and receive tasks</p> <p>1'b1: SPI is in the process of sending or receiving</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b0
[11:6]	RO	<p>RX FIFO fill level</p> <p>The amount of data in the Rx FIFO, in bytes</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	6'h0
[5:0]	RO	<p>Tx FIFO fill level</p> <p>The amount of data in the Tx FIFO, in bytes</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	6'h0

14.4.9 SPI Timeout Register

Table 117 SPI Timeout Register

bit access		Instructions	reset value
[31]	RW	<p>spi_timer_en</p> <p>1'b0: Do not allow timer timing</p> <p>1'b1: allow timer timing</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	1'b0
[30:0]	RW	<p>SPI_TIME_OUT</p> <p>When a transmission is completed, in the receiving channel rxfifo, if the data at the end cannot trigger the receiving interrupt</p> <p>When RxFifoRdy or DMA request, a timing mechanism needs to be used to notify the CPU to remove the end data.</p> <p>Specific method: When rxfifo is in idle state (no read and write operations, no dma request, cs is invalid, there is data in rxfifo, and the amount of data is less than or equal to the RxTrigger level), start counting and reach the setting of this register.</p> <p>If the value is set, the timeout interrupt is triggered, and the CPU is requested to remove the end data.</p> <p>Any read or write to rxfifo will clear the timeout timer.</p> <p>The time represented is: $T = \text{SPI_TIME_OUT}/\text{FAPB_CLK}$</p> <p>Note: Both master/slave are valid Motorola/TI/microwire mode is valid</p>	31'h0

14.4.10 Data Transmission Register

Table 118 SPI data transmission register

bit access		Instructions	reset value

[31:0] RW		<p>Window address for writing data to Tx FIFO</p> <p>Note:</p> <p>Both master/slave are valid</p> <p>Motorola/TI/microwire mode is valid</p>	32'h0
-----------	--	--	-------

14.4.11 Transfer Mode Register

Table 119 SPI transfer mode register

bit access		Instructions	reset value
[31:30]		RSV	16'd0
[29:24] RW		<p>TI_BLK_LEN</p> <p>In the timing mode of TI, the length of each block transmission, that is, the transmission data length after each CS is valid.</p> <p>Support 4~32bit</p> <p>6'h4: 4bit long data</p> <p>6'h5: 5bit long data</p> <p>6'h6: 6bit long data</p> <p>.....</p> <p>6'h20: 32bit long data</p> <p>Note: master is valid</p> <p>TI mode active</p>	6'd0
[16]	RW	<p>MICRO_BURST</p> <p>1b'1: In Microwire mode, burst transmission is used, that is, Tx sends control words, Rx receives data, and then</p> <p>Alternately, MICRO_CONTROL_LEN indicates the length of the control word, MICRO_DAT_LEN indicates</p>	1'b0

		<p>is the length of the sent or received word, Tx/Rx length indicates the effective sck, burst during the entire transmission process</p> <p>In mode, the number of times of sending and receiving alternately is $(Tx/Rx\ length)/(MICRO_CONTROL_LEN+MICRO_DAT_LEN+1)$</p> <p>1'b0: In Microwire mode, burst transmission is not used</p> <p>In this mode, there are two cases</p> <p>1) tx_ch_on = 1, rx_ch_on = 0, at this time, only sending, MICRO_CONTROL_LEN means Control word length, Tx/Rx length indicates the valid sck during the entire transmission process, the length of the data sent at this time The degree is $m*MICRO_DAT_LEN = Tx/Rx\ length - MICRO_CONTROL_LEN$, where m represents How many (MICRO_DAT_LEN) length words to send</p> <p>2) tx_ch_on = 1, rx_ch_on = 1, at this time, Tx sends control word, Rx receives data, MICRO_CONTROL_LEN indicates the length of the control word, and Tx/Rx length indicates the entire transmission Valid sck in the process, the length of the received data is $m*MICRO_DAT_LEN = Tx/Rx\ length$ MICRO_CONTROL_LEN-1, where m indicates how many (MICRO_DAT_LEN) length words are received Note: master is valid microwire mode works</p>	
[13:8] RW		<p>MICRO_DAT_LEN</p> <p>In Microwire mode, in burst mode, the length of each burst transmission data</p> <p>From 1 to 32:</p> <p>6'h1: 1bit long data</p> <p>6'h2: 2bit long data</p>	6'd0

		<p>6'h3: 3bit long data</p> <p>.....</p> <p>6'h20 32bit long data</p> <p>Note: master is valid</p> <p>microwire mode works</p>	
[5:0] RW		<p>MICRO_CONTROL_LEN</p> <p>In Microwire mode, the length of the command word</p> <p>From 1 to 32:</p> <p>6'h1: 1bit long command</p> <p>6'h2: 2bit long command</p> <p>6'h3: 3bit long command</p> <p>.....</p> <p>6'h20 32bit long command</p> <p>Note: master is valid</p> <p>microwire mode works</p>	6'd0

14.4.12 Data Length Register

Table 120 SPI Data Length Register

bit access		Instructions	reset value
[31:16] RO		<p>When used as a slave, the length of the data sent out during the valid period of cs, the unit is bit</p> <p>Note: slave is valid</p> <p>Motorola mode is valid</p>	16'h0

[15:0] RO		When used as a slave, during the valid period of cs, the length of the received data, the unit is bit Note: slave is valid Motorola mode is valid	16'h0
-----------	--	---	-------

14.4.13 Data Receive Register

Table 121 SPI Data Receive Register

bit access		Instructions	reset value
[31:0] RO		Window address for reading data from Rx FIFO Note: Both master/slave are valid Motorola/TI/microwire mode is valid	32'h0

15 I2C Controller

15.1 Function overview

The I2C bus is a simple, bidirectional two-wire synchronous serial bus. It requires only two wires to transmit information between devices connected to the bus interest.

The master device is used to start the bus to transmit data and generate a clock to open the device for transmission. At this time, any addressed device is considered a slave.

piece. The relationship between master and slave, sending and receiving on the bus is not constant, but depends on the direction of data transfer at this time. If the master wants to send data to the slave device, the host first addresses the slave device, then actively sends data to the slave device, and finally terminates the data transfer by the host; if the host wants to receive

The data of the slave device is first addressed by the master device. Then the host receives the data sent from the device, and finally the host terminates the receiving process. in this case. The host is responsible for generating the timing clock and terminating the data transfer.

15.2 Main Features

- ÿ APB bus protocol standard interface
- ÿ Can only be used as a master device controller
- ÿ I2C working rate can be configured, 100KHz~400KHz
- ÿ Multiple GPIO can be multiplexed into I2C communication interface
- ÿ Quickly output and detect timing signals

15.3 Functional Description

15.3.1 Transmission rate selection

The data transfer rate on the I2C bus can be configured from 100KHz to

Any bus frequency integer divide value between 400KHz.

15.3.2 Interrupt and start-stop controllable

Enable or disable the I2C controller to generate interrupts by setting Bit6 of the register CTR, and can also start at any time by setting Bit7

Or stop the work of the I2C controller.

15.3.3 Fast output and detection signal

By setting the corresponding bits of the register CR_SR, the controller can quickly output or detect the bus START signal, the bus STOP signal, and the total

Line ACK signal, bus NACK signal. In master mode, the I2C interface initiates data transfers and generates clock signals, a serial data transfer

The output always starts with a start signal and ends with a stop signal. Once the start signal is generated on the bus, the master mode is selected.

15.4 Register Description

15.4.1 Register List

Table 122 I2C register list

offset address	name	abbreviation	access	describe	reset value
0X0000	Clock divider register_1	PRERlo	RW stores the frequency division value of the lower 8 bits, which is used for APB	The bus clock is divided	0X0000_00FF
0X0004	Clock divider register_2	PRERhi	RW stores the frequency division value of the upper 8 bits, which is used for APB	The bus clock is divided	0X0000_00FF
0X0008	Control register	CTR	RW is used to control interrupt enable and I2S control	enable	0X0000_0040
0X000C	Data register	TXR_RXR	RW is used to store the data to be sent or receive	0X0000_0000	

				The data	
0X0010	Transceiver Control Register	CR_SR	RW is used to control some data read and write related operations	0X0000_0000	
0X0014	TXR read register	TXR	RO reads the TXR register value 0X0000_0000 when I2C is transmitting		
0X0018	CR read register	CR	RO Read the set value of the I2C control register CR	0X0000_0000	

15.4.2 Clock divider register_1

Table 123 I2C clock divider register_1

bit access		Instructions	reset value
[31:8]		reserve	
[7 : 0] RW		<p>The clock divider configures the lower 8bits of prescale.</p> <p>E.g:</p> <p>apb_clk=40MHz, SCL=100KHz</p> $\text{prescale} = (40*1000)/(5*100) - 1 = 16'd79$ <p>apb_clk = 40M, SCL=400K</p> $\text{prescale} = (40*1000)/(5*400) - 1 = 16'd19$	8'hff

15.4.3 Clock divider register_2

Table 124 I2C clock divider register_2

bit access		Instructions	reset value
[31:8]		reserve	
[7 : 0] RW		<p>The clock divider configures the high 8bits of prescale.</p> <p>E.g:</p>	8'hff

		<p>apb_clk=40MHz, SCL=100KHz</p> <p>prescale = $(40*1000)/(5*100) - 1 = 16'd79$</p> <p>apb_clk = 40M, SCL=400K</p> <p>prescale=(40*1000)/(5*400) - 1 = 16'd19</p>	
--	--	--	--

15.4.4 Control Register

Table 125 I2C Control Register

bit access		Instructions	reset value
[31:8]		reserve	
[7]	RW	I2C enable control, 1'b0: Disable 1'b1: enable	1'b0
[6]	RW	interrupt MASK, 1'b0: Enable interrupt generation 1'b1: Interrupt generation is not allowed	1'b1
[5:0]		reserve	

15.4.5 Data Register

Table 126 I2C data register

bit access		Instructions	reset value
[31:8]		reserve	
[7:0] WR		When writing this register, it is the transmit register TXR,	8'h0

		<p>represents the next byte to be sent,</p> <p>When it is a device address,</p> <p>[0]: 1 means read, 0 means write.</p> <p>When reading this register, it is the receiving register RXR,</p> <p>is the latest byte received from I2C.</p>	
--	--	--	--

15.4.6 Transceiver Control Register

Table 127 I2C Transceiver Control Register

bit access		Instructions	reset value
[31:8]		reserve	
[7:0] WR		<p>When writing this register, it is CR, and the function is as follows:</p> <p>[7]: STA, control to generate START timing; 1'b0: invalid 1'b1: Generate START timing</p> <p>[6]: STO, control generates STOP sequence; 1'b0: invalid 1'b1: Generate STOP timing</p> <p>[5]: RD, read from SLAVE; 1'b0: invalid</p>	8'h0

		<p>1'b1: read from SLAVE</p> <p>[4]: WR, write to SLAVE;</p> <p>1'b0: invalid</p> <p>1'b1: write to SLAVE</p> <p>[3]: Control returns ACK/NACK to SLAVE;</p> <p>1'b0: return ACK</p> <p>1'b1: Back to NAK</p> <p>[2:1]: reserved;</p> <p>[0]: IACK, clear the interrupt status, 1 is valid;</p> <p>1'b0: invalid</p> <p>1'b1: clear interrupt flag</p> <p>When reading this register, it is SR, and the function is as follows:</p> <p>[7]: RxACK, ACK/NACK status received from SLAVE;</p> <p>1'b0: ACK received from SLAVE</p> <p>1'b1: NAK received from SLAVE</p> <p>[6]: BUSY;</p> <p>1'b0: 0 is set after STO</p>	
--	--	---	--

		<p>1'b1: Set 1 after STA</p> <p>[5]: AL, Arbitration Lost, this bit is reserved;</p> <p>[4:2]: reserved;</p> <p>[1]: TIP;</p> <p>1'b0: No transfer in progress</p> <p>1'b1: there is a transfer in progress</p> <p>[0]: IF, interrupt status bit;</p> <p>1'b0: No interrupt is generated</p> <p>1'b1: Set to 1 when transfer is complete or AL</p>	
--	--	--	--

15.4.7 TXR Readout Register

Table 128 I2C TXR readout register

bit access		Instructions	reset value
[31:8]		reserve	
[7:0]	RO	Read only, read value of TXR register, See TXR_RXR register for function description;	8'h0

15.4.8 CR read register

Table 129 I2C CR read register

bit access		Instructions	reset value
------------	--	--------------	-------------

[31:8]		reserve	
[7:0]	RO	Read only, read value of CR register, See CR_SR register for function description;	8'h0

16 I2S Controller

16.1 Function overview

I2S (Inter-IC Sound) is an audio system for digital audio equipment (such as CD players, digital sound processors, and digital TV sound systems).

A bus standard developed for the transmission of audio data between. It adopts the design of independent wires to transmit clock and data signals.

The data is separated from the clock signal, which avoids the distortion induced by the time difference, and saves the user the cost of purchasing professional equipment that resists audio jitter. mark

A standard I2S bus cable consists of 3 serial conductors: 1 is a time division multiplexed (TDM) data line; 1 is a word select line; 1 is the clock line.

16.2 Main Features

- ÿ Implement I2S interface, support I2S and PCM protocols
- ÿ Support amba APB bus interface, 32bit single read and write operations
- ÿ Support master-slave mode
- ÿ Support 8, 16, 24, 32 bit width, the highest sampling frequency is 192KHz
- ÿ Support mono and stereo mode
- ÿ Compatible with I2S and MSB justified data format, compatible with PCM A/B format
- ÿ Support DMA request read and write operations, only support word-by-word operations

16.3 Functional Description

16.3.1 Multiple Mode Support

By setting Bit[25:24] of the I2S Control register, the data format can be set to I2S format, MSB justified format, PCM A

Format, or PCM B format; by setting Bit[22] of the I2S Control register, mono or stereo mode can be selected. pass

Setting Bit[5:4] of the I2S Control register can set the bit width of the data transmission word, which can be set to 8bit, 16bit, 24bit, 32bit.

16.3.2 Zero Crossing Detection

By setting Bit[17:16] of the I2S Control register, you can set whether to enable the zero-cross detection function of the left and right channels; by setting

Bit[9:8] of the I2S_IMASK register can set whether the left and right channel zero-cross detection function generates an interrupt. If detection is enabled, and

And the interrupt is enabled, when the zero-crossing phenomenon is detected, the program will execute the interrupt subroutine, and at the same time Bit[9:8] of the I2S_INT_FLAG register

The corresponding bit will be set to 1.

16.3.3 Efficient data transfer

The FIFO memory is a first-in-first-out dual-port buffer, that is, the first data entered into it is the first to be shifted out, and one of the memory's

The input port, and the other port is the output port of the memory. The I2S controller integrates two (one for each transceiver) FIFO storage with a depth of 8 words

It can increase the data transfer rate, handle a large number of data streams, and match systems with different transfer rates, thereby improving system performance. able to pass

Setting Bit[14:12] and Bit[11:9] of the I2S Control register can set the trigger level of RXFIFO and TXFIFO to meet the

Performance requirements at different transfer rates. After the trigger level of the FIFO is triggered, an interrupt or DMA can be triggered to transfer the data from the internal

Move data to TXFIFO or move data from RXFIFO to memory.

16.4 I2S/PCM Timing Diagram

This module provides support for 4 protocols, standard I2S, MSB Justified, PCM-A, PCM-B. Through the configuration register 0x00[25:

24] to choose which protocol to use. The specific interface timing of each protocol is shown in Fig1 to Fig4.

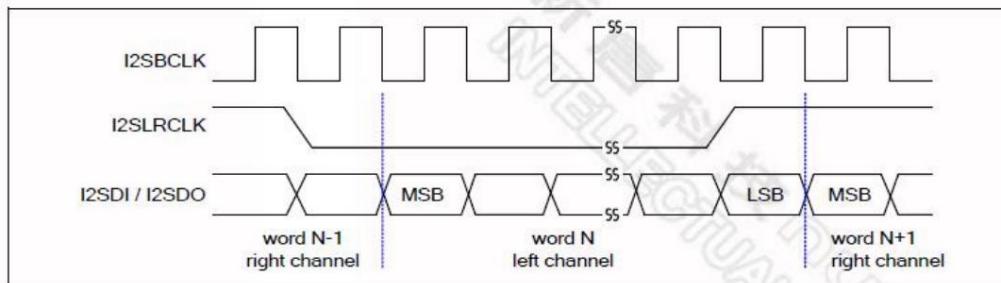


Fig1. I2S Bus Timing Diagram (PCM=0, Format=0)

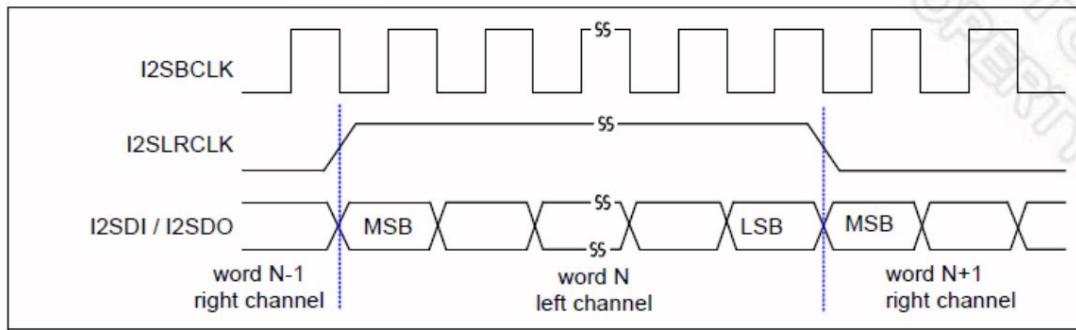


Fig2. MSB Justified Timing Diagram (PCM=0, Format=1)

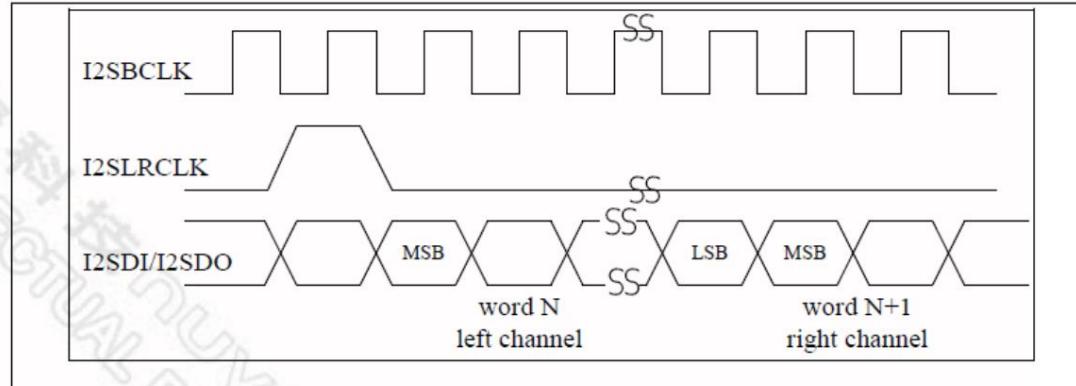


Fig3. PCM A Audio Diagram (PCM=1, Format=0)

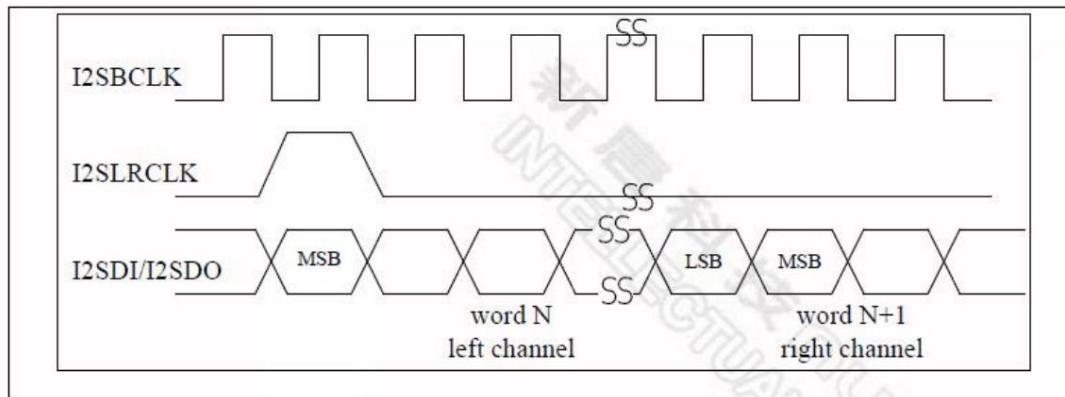


Fig4. PCM B Audio Diagram (PCM=1, Format=1)

16.5 FIFO storage structure diagram

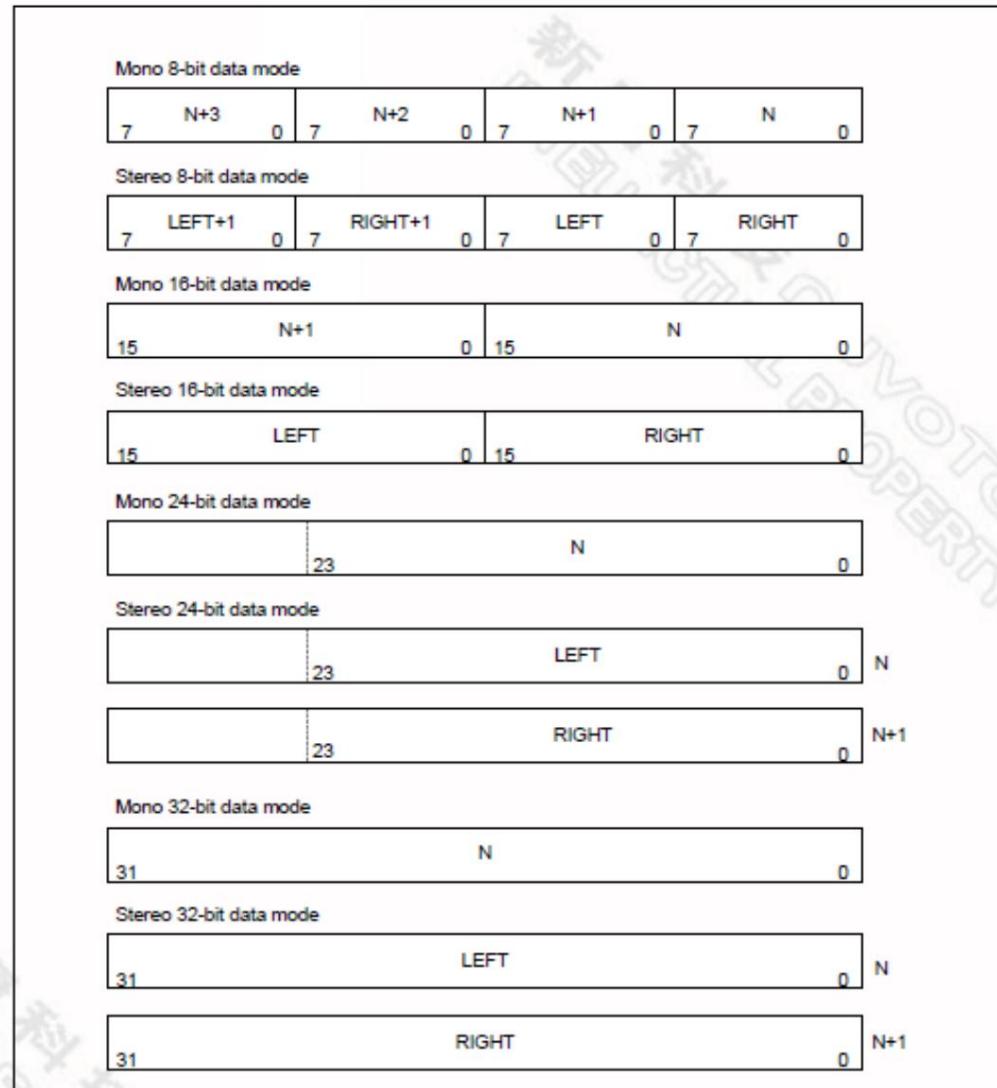


Fig 5. FIFO storage structure

The I2S module provides two 8x32bit FIFOs, one as TX FIFO and one as RX FIFO.

The storage structure of the data in the FIFO varies according to the working mode. When working in mono 8bit mode, each word in the FIFO

Four sample data can be stored. When working in dual-channel 8bit mode, the left and right channel data are alternately stored in the FIFO, at low

The byte of the bit stores the data of the right channel, and the byte in the high order stores the data of the left channel. A word can store 2 samples

according to. In simplex 16bit mode, one word can store 2 samples, in duplex 16bit mode, one word can store one sample

Sample data, the lower 16 bits are the right channel, and the upper 16 bits are the left channel. In 24bit and 32bit mode, each word can only store one

Sample data or data of one channel, the specific storage method is shown in Figure 5.

16.6 I2S module working clock configuration

The working clock configuration of the I2S module can be configured by setting the 0x40000718 register in the clock and reset module to select the external clock source.

The clock is still the internal 160MHz clock, whether the MCLK clock is turned on, and the operating frequencies of MCLK and BCLK.

By setting register 0x40000718[0], you can choose to use the internal 160MHz clock or use the external clock as the I2S module clock

source. If an external clock is selected, the I2S module will use the clock input from IO- I2S_M_EXTCLK (PA_5, Option 4) as the module clock

Zhongyuan.

And 0x40000718[1] selects whether to turn on the MCLK clock. Bits[7:2] are the area to configure the MCLK divider ratio. Calculated as follows:

$$F_{mclk} = F_{I2SCLK} / MCLKDIV$$

F_{mclk} is the actual MCLK frequency;

F_{I2SCLK} is the clock source of the I2S module. If internal clock is selected, $F_{I2SCLK} = 160MHz$; if external clock is selected, then

F_{I2SCLK} is equal to the external input clock frequency;

MCLKDIV is the clock divider configured by Register 0x40000718[7:2]. It should be noted that $MCLKDIV \geq 2$;

Bits[17:8] of register 0x40000718 is the area for configuring the divider ratio of the BCLK clock. The formula for calculating the frequency division ratio is as follows:

$$F_{BCLK} = F_{I2SCLK} / BCLKDIV$$

F_{BCLK} is the actual working frequency of BCLK;

F_{I2SCLK} is the clock source of the I2S module. If internal clock is selected, $F_{I2SCLK} = 160MHz$; if external clock is selected, then

F_{I2SCLK} is equal to the external input clock frequency;

BCLKDIV is the frequency division ratio that needs to be configured. Different frequency division ratios need to be selected according to different working modes. The formula for selecting the divider ratio is as follows:

$$BCLKDIV = \text{round}(F_{I2SCLK} / (Fs * W * F))$$

Fs is the sampling frequency of audio data on the I2S interface, up to 192KHz;

W is the sampling bit width, 8/16/24/32 bit can be selected;

F is for mono/dual channel selection. F=1 when the transmission data is mono, and F=2 when the transmission data is two channels;

The final calculated divider ratio is rounded up.

The following is an example of selecting BCLKDIV according to the actual working mode:

If the internal clock is selected as the module clock source, 128KHz sampling rate and 24bit dual-channel data need to be transmitted, then the calculation needs to set BCLKDIV

The process is as follows:

BCLKDIV=round(160*10e6/128*10e3*24*2)=10'd26;

The description of the frequency divider register is as follows:

0x18	I2S_Clk_Ctrl	[0]	RW EXTAL_EN	External clock select Select whether use External or Internal clock for I2S block 0=internal clk 1=external clk Note: When External clock is enabled, the external clk must be $2^N \times 256$ fs, where fs is sample frequency and N must be integer.	1'b0
------	--------------	-----	-------------	---	------

	[1]	RW MCLKEN	MCLK enable 0=MCLK disabled 1=MCLK enabled	1'b0
	[7:2]	RW MCLKDIV	MCLK divider If external clock is selected, this divider is used to produce proper MCLK frequency. $F_{mclk} = F_{I2SCLK}/MCLKDIV$ Where $MCLKDIV \geq 2$ $F_{mclk} = F_{I2SCLK}$ when $MCLKDIV = 0$ Where F_{I2SCLK} is external clk. Note: F_{mclk} should be configured as 256 * fs where fs is sample frequency.	6'd0
	[17:8]	RW BCLOCKDIV	BCLK divider $F_{BCLK} = F_{I2SCLK}/BCLOCKDIV$ Note: When EXTAL_EN is not selected, Internal PLL is used and $F_{I2SCLK} = 160MHz$	10'd0

				<p>When WXTAL_EN is enabled,</p> <p>$F_{I2SCLK} = \text{External crystal frequency}$</p> <p>$BCLKDIV = \text{round}(F_{I2SCLK}/(Fs * W * F))$</p> <p>Where F_s is sample frequency of audio data and W is word width.</p> <p>$F=2$ when data is stereo and</p> <p>$F=1$ when data is mono.</p> <p>For example, if internal PLL is used and the data width is 24bit, Format is stereo format, sample frequency is 128KHz.</p> <p>Then the BCLKDIV should be configured as $(160 * 10e6 / 128 * 10e3 * 24 * 2) = 10'd26$</p>	
	[31:18] RO			RSV	14'd0

16.7 Other function description:

16.7.1 Zero-crossing detection:

To avoid noise caused by sudden frequency changes due to data corruption, the I2S module provides zero-crossing detection for each channel.

When the transmitted adjacent two data symbol bits change, the module will generate an interrupt to remind the MCU to detect and process;

Data is forcibly muted.

16.7.2 Mute function

When the mute function is turned on, the data will still be sent, but the output data will be forced to 0;

16.7.3 Interrupts

This module provides transmit/receive completion interrupt, left and right channel zero-crossing detection interrupt, transmit/receive FIFO threshold interrupt (number of data in transmit FIFO)

If the data is lower than the threshold, the data in the receive FIFO is higher than the threshold), the underflow/overflow interrupt of the transmit/receive FIFO. interrupted state

Polled in register 0x08.

16.7.4 FIFO status query

Register 0x10 provides the status query function of the CPU to the transmit/receive FIFO in the I2S module. Through the register CPU can check the FIFO

How much data is left in unprocessed. There are several bytes in the last word in the receive FIFO that are valid data.

16.8 Data transfer process

16.8.1 The master sends audio data

1. Configure the used pins, SDO, BCLK, LRCLK

2. Refer to Section 2.4 to set register 0x40000718 in the clock and reset module, and configure the working clock frequency;

3. Set I2S register 0x00, set to master mode, configure transmission format, channel selection, data bit width, left and right channels are

No Enable zero-crossing detection, and set the transmit path -txen to be enabled.

4. Set register 0x4 to enable the interrupt you want to use;

5. If DMA is used to transmit data, select the channel to be used in the DMA module, and configure the address and length of the transmitted data. and in I2S

The DMA is enabled in 0x0 in the module register, and the FIFO threshold is set. DMA is automatically requested when the data in the FIFO falls below the threshold range

Transport data.

6. Write the data to be transmitted to the transmit FIFO address - register 0x10, enable register 0x0[0], the module will automatically remove the data from the FIFO

Take out the data and send it to the I2S bus.

7. When the data in the FIFO is less than the set threshold, the module will request data from the DMA module, or send a TXTHIF interrupt.

8. When all the data in the FIFO is taken out, the TXDONE interrupt will be set. When the transmission of the last frame is completed, the TXUDIF interrupt will be set.

When the CPU transmission is completed, the module stops transmission.

16.8.2 Receiving audio data from the slave

1. Configure the used pins, SDI, BCLK, LRCLK

2. Set the I2S register 0x00 to slave mode to configure the transmission format, channel selection, data bit width, whether the left and right channels are open

Enable zero-crossing detection, and set receive channel -rxen on.

3. Set register 0x4 to enable the interrupt you want to use;

4. If using DMA to transmit data, select the channel to be used in the DMA module, and configure the address and length of the transmitted data. and in I2S

The DMA is enabled in 0x0 in the module register, and the FIFO threshold is set. When the data is above the threshold range, it will automatically request DMA transfer data.

5. Enable register 0x0[0], after the module detects valid BCLK and LRCLK, it will automatically collect data from SDI and store it in FIFO middle. When the data in the FIFO is higher than the set threshold, the module will request the DMA to transfer the data to the memory, or send the RXTHIF break. The RXDONE interrupt is generated when the CPU turns off RXEN. The CPU can query how much is in the receive FIFO through register 0x10 data, how many bytes of valid data in the last word. The last remaining data is then processed according to the FIFO status.

16.8.3 The master receives audio data

1. Configure the hardware used, SDI, SCLK, LRCLK
2. Refer to Section 2.4 to set register 0x40000718 in the clock and reset module, and configure the working clock frequency;
3. Set the I2S register 0x00 to master mode, configure the transmission format, channel selection, data bit width, whether the left and right channels are on
Enable zero-crossing detection, and set the receive channel rxen to be on.
4. Set register 0x4 to enable the interrupt you want to use;
5. If DMA is used to transmit data, select the channel to be used in the DMA module, and configure the address and length of the transmitted data. and in I2S

The DMA is enabled in 0x0 in the module register, and the FIFO threshold is set. When the data is above the threshold range, it will automatically request DMA transfer data.

6. Enable register 0x0[0], the module will automatically send BCLK and LRCLK, and collect data from SDI and store it in FIFO. when
When the data in the FIFO is higher than the set threshold, the module will request the DMA to transfer the data to the memory, or send the RXTHIF interrupt. when
The RXDONE interrupt is generated when the CPU turns off RXEN. The CPU can query how much data is in the receive FIFO through register 0x10,
The number of bytes of valid data in the last word. The last remaining data is then processed according to the FIFO status.
7. Turn off the i2s enable after data reception is complete.

16.8.4 Slave sending audio data

1. Configure the used pins, SDO, BCLK, LRCLK

2. Set the I2S register 0x00 to slave mode, configure the transmission format, channel selection, data bit width, whether the left and right channels are

Enable zero-crossing detection, and set the transmit path txen to be on.

3. Set register 0x4 to enable the interrupt you want to use;

4. If DMA is used to transmit data, select the channel to be used in the DMA module, and configure the address and length of the transmitted data. and in I2S

The DMA is enabled in 0x0 in the module register, and the FIFO threshold is set. DMA is automatically requested when the data in the FIFO falls below the threshold range

Transport data.

5. Write the data to be transmitted to the transmit FIFO address-register 0x10, enable register 0x0[0], when the module detects a valid BCLK

and LRCLK, the module will automatically take out the data from the FIFO and send it to the SDO.

6. When the data in the FIFO is less than the set threshold, the module will request data from the DMA module, or send a TXTHIF interrupt.

7. When all the data in the FIFO is taken out, the TXDONE interrupt will be set. When the transmission of the last frame is completed, the TXUDIF interrupt will be set.

When the CPU transmission is completed, the module stops transmission.

16.8.5 Full Duplex Mode

1. Configure the used pins, SDI, SDO, BCLK, LRCLK

2. If it is master mode, you need to configure the clock frequency division.

3. Set I2S register 0x00, configure working mode (master/slave), configure transmission format, channel selection, data bit width, left and right channels are

No Enable zero-crossing detection, and set the transmit path txen and receive path rxen to be enabled.

4. Set register 0x4 to enable the interrupt you want to use;

5. If DMA is used to transmit data, select the channel to be used in the DMA module, and configure the address and length of the transmitted data. and in I2S

The DMA is enabled in 0x0 in the module register, and the FIFO threshold is set. DMA is automatically requested when the data in the FIFO falls below the threshold range

Transport data.

6. Write the data to be transmitted to the transmit FIFO address - register 0x10

7. If it is master mode, enable register 0x0[0], the module will start to transmit BCLK and LRCLK, and fetch data from transmit FIFO.

The data starts to be sent out from the SDO port, and the data received from SDI is stored in the receive FIFO at the same time.

8. If it is in slave mode, when the module detects valid BCLK and LRCLK, the module will automatically fetch data from the transmit FIFO

It is sent to SDO, and the data received from SDI is stored in the receive FIFO at the same time.

9. When the data in the transmit FIFO is less than the set threshold, the module will request data from the DMA module, or send a TXTHIF interrupt.

10. When the data in the FIFO is higher than the set threshold, the module will request the DMA to transfer the data to the memory, or send the RXTHIF interrupt.

11. When all the data in the TXFIFO is taken out, the TXDONE interrupt will be set. When the transmission of the last frame is completed, the TXUDIF interrupt will be set.

Notify the CPU that the sending is completed, and the module stops sending.

12. The RXDONE interrupt is generated when the CPU turns off RXEN. The CPU can query how many numbers are in the receive FIFO through register 0x10

According to the number of bytes of valid data in the last word. The last remaining data is then processed according to the FIFO status.

16.9 Register Description

16.9.1 Register List

Table 130 I2S register list

offset address name		abbreviation	access	describe	reset value
0X0000 Control Register	I2S Control	RW/RO	controls I2S related functions, see the following chapters for details;		0X0000_4800
0X0004 Interrupt mask register I2S_IMASK		RW	controls to turn on or off all interrupts in I2S		0X0000_03FF
0X0008 Interrupt Flag Register I2S_INT_FLAG		RW/RO	flag bit, which can be used to query whether an interrupt is generated and Clear related interrupts		0X0000_0000
0X000c Status Register	I2S_STATUS	RO	is used to query the related status of FIFO during I2S communication state		0X0000_0000

0X0010	Data transmission register I2S_TX		The WO controller will send the data in it to the bus 0X0000_0000	
0X0014	Data receiving register I2S_RX		The RO controller will receive the data on the bus into it 0X0000_0000	

16.9.2 Control Register

Table 131 I2S Control Register

bit access		Instructions	reset value
[31:29] RO		RSV	7'h0
[28]	RW	Mode selection 0 = master mode 1 = slave mode	1'b0
[27]	RW	Duplex mode selection 1 = Duplex mode enabled 0 = Disable duplex mode	1'b0
[26]	RW	Timeout count control bit, when this bit is set to 1 and the transmission process is forcibly stopped by the master device, no reception will occur Done (RXDONE) interrupt	1'h0
[25:24] RW		FORMAT Data format selection 2'b00: I2S data format 2'b01: MSB Justified Data Format 2'b10: PCM A sound data format 2'b11: PCM B sound data format	2'b0
[twenty three]	RW	RXLCH	1'b0

		<p>Channel receive enable control bit</p> <p>1'b0: Enable to receive right channel data</p> <p>1'b1: Enable to receive left channel data</p> <p>Note: This bit is only valid when the mono mode of MONO_STEREO is selected</p>	
[twenty two]	RW	<p>MONO_STEREO</p> <p>Mono Stereo Select Bits</p> <p>1'b0: Data is transmitted in stereo format</p> <p>1'b1: data is transmitted in mono format</p>	1'b0
[twenty one]	RW	<p>RXDMAEN</p> <p>Receive DMA request enable bit</p> <p>1'b0: Disable send DMA request</p> <p>1'b1: Enable send DMA request</p> <p>Note: When the transfer DMA request is enabled and the number of words in the RXFIFO is equal to or greater than RXTH, the I2S controller will send a transfer request to the DMA and stop the DMA transfer until the RXFIFO is empty.</p>	1'b0
[20]	RW	<p>TXDMAEN</p> <p>Send DMA request enable bit</p> <p>1'b0: Disable send DMA request</p> <p>1'b1: Enable send DMA request</p> <p>Note: When the transmit DMA request is enabled and the number of words in the TXFIFO is less than TXTH, the I2S controller will issue a transfer request to the DMA until the TXFIFO is full and the DMA transfer will not be stopped.</p>	1'b0
[19]	WO	<p>RXCLR</p> <p>clear RXFIFO</p>	1'b0

		<p>1'b0: invalid</p> <p>1'b1: clear RXFIFO</p> <p>Note: Write 1 to clear the RXFIFO, which is automatically cleared by hardware. Reading this bit always returns 0</p>	
[18] WO		<p>TXCLR</p> <p>clear TXFIFO</p> <p>1'b0: invalid</p> <p>1'b1: clear TXFIFO</p> <p>Note: Write 1 to clear the TXFIFO, which is automatically cleared by hardware. Reading this bit always returns 0</p>	1'b0
[17]	RW	<p>LZCEN</p> <p>Left channel zero-crossing detection enable control bit</p> <p>1'b0: Stop left channel zero-crossing detection</p> <p>1'b1: Enable left channel zero-crossing detection</p>	1'b0
[16]	RW	<p>RZCEN</p> <p>Right channel zero-crossing detection enable control bit</p> <p>1'b0: Stop right channel zero-crossing detection</p> <p>1'b1: Enable right channel zero-crossing detection</p>	1'b0
[15]	RW	<p>Rx_clk_phase_sel</p> <p>Receive clock phase selection</p> <p>1'b0: Default mode</p> <p>Shown with the I2S bus timing mentioned above</p> <p>1'b1: Invert mode</p> <p>Shown as a reversal of the I2S bus timing mentioned above</p>	1'b0

[14:12] RW		<p>RXTH</p> <p>RXFIFO threshold</p> <p>3'b000: Set the threshold to 0 words</p> <p>3'b000: Set the threshold to 1 word</p> <p>...</p> <p>3'b111: Set the threshold to 7 words</p> <p>Note: The RXTHIF bit is set when the existing word in the RXFIFO is equal to or more than the value of RXTH. this</p> <p>You can choose to trigger RXDMA or I2S interrupt according to the settings</p>	3'h4
[11:9] RW		<p>TXTH</p> <p>TXFIFO threshold</p> <p>3'b000: Set the threshold to 0 words</p> <p>3'b000: Set the threshold to 1 word</p> <p>...</p> <p>3'b111: Set the threshold to 7 words</p> <p>Note: The TXTHIF bit is set when the existing word in the TXFIFO is equal to or less than the value of TXTH. this</p> <p>You can choose to trigger TXDMA or I2S interrupt according to the settings</p>	3'h4
[8]	RW	<p>Tx_clk_phase_sel</p> <p>Select transmit clock phase mode</p> <p>1'b0: Default mode</p> <p>Shown with the I2S bus timing mentioned above</p> <p>1'b1: Invert mode</p> <p>Shown as a reversal of the I2S bus timing mentioned above</p>	1'b0

[7:6] RW		RSV	2'h0
[5:4] RW		<p>WDWIDTH</p> <p>Transfer word length setting bits</p> <p>2'b00: word length 8 bits</p> <p>2'b01: word length 16 bits</p> <p>2'b10: word length 24 bits</p> <p>2'b11: word length 32 bits</p>	2'b0
[3]	RW	<p>MUTE</p> <p>Transmit mute enable flag</p> <p>1'b0: transfer data from shift register, normal operation mode</p> <p>1'b1: Set transmit data to 0, mute the sound</p>	1'b0
[2]	RW	<p>RXEN</p> <p>receive enable flag</p> <p>1'b0: Stop I2S data reception</p> <p>1'b1: Enable I2S data reception</p>	1'b0
[1]	RW	<p>TXEN</p> <p>transfer enable flag</p> <p>1'b0: Stop I2S data transmission</p> <p>1'b1: Enable I2S data transfer</p>	1'b0
[0]	RW	<p>I2SEN</p> <p>I2S enable flag</p> <p>1'b0: Disable</p>	1'b0

		1'b1: enable	
--	--	--------------	--

16.9.3 Interrupt Mask Register

Table 132 I2S Interrupt Mask Register

bit access		Instructions	reset value
[31:10] RO		RSV	22'h0
[9]	RW	<p>LZCIMASK</p> <p>Left channel zero crossing interrupt enable flag</p> <p>1'b0: Interrupt disabled</p> <p>1'b1: Interrupt enable</p> <p>An interrupt is generated when the interrupt is enabled and a zero crossing is detected on the left channel</p>	1'b1
[8]	RW	<p>RZCIMASK</p> <p>Right channel zero crossing interrupt enable flag</p> <p>1'b0: Interrupt disabled</p> <p>1'b1: Interrupt enable</p> <p>An interrupt is generated when the interrupt is enabled and a zero crossing is detected on the right channel</p>	1'b1
[7]	RW	<p>TXDONEMASK</p> <p>Transmit complete interrupt enable bit</p> <p>1'b0: Interrupt disabled</p> <p>1'b1: Interrupt enable</p> <p>An interrupt is generated when the interrupt is enabled and the TXFIFO is empty</p>	1'b1
[6]	RW	TXTHIMASK	1'b1

		<p>TXFIFO threshold interrupt enable bit</p> <p>1'b0: Interrupt disabled</p> <p>1'b1: Interrupt enable</p> <p>An interrupt is generated when the interrupt is enabled and the number of data in the TXFIFO is equal to or less than TXTH</p>	
[5]	RW	<p>TXOVIMASK</p> <p>TXFIFO overflow interrupt enable bit</p> <p>1'b0: Interrupt disabled</p> <p>1'b1: Interrupt enable</p> <p>Note: When the interrupt is enabled, the TXFIFO is full, and the CPU writes data to the TXFIFO, the TXOVIF flag will be set</p>	1'b1
[4]	RW	<p>TXUDIMASK</p> <p>TXFIFO underflow interrupt enable bit</p> <p>1'b0: Interrupt disabled</p> <p>1'b1: Interrupt enable</p> <p>Note: When the TXFIFO underflow interrupt is enabled and TXUDIF is detected as 1, an underflow interrupt will be generated</p>	1'b1
[3]	RW	<p>RXDONEMASK</p> <p>Receive complete interrupt enable flag bit</p> <p>1'b0: Interrupt disabled</p> <p>1'b1: Interrupt enable</p> <p>When the reception completion interrupt is enabled and the reception process is completed, the reception completion interrupt will be generated</p>	1'b1
[2]	RW	<p>RXTHIMASK</p> <p>RXFIFO threshold interrupt enable flag</p>	1'b1

		<p>1'b0: Interrupt disabled</p> <p>1'b1: Interrupt enable</p> <p>Generated when the RXFIFO threshold interrupt is enabled and the number of data in the RXFIFO is equal to or more than the threshold</p> <p>Generate RX interrupt</p>	
[1]	RW	<p>RXOVIMASK</p> <p>RXFIFO overflow interrupt enable bit</p> <p>1'b0: Interrupt disabled</p> <p>1'b1: Interrupt enable</p> <p>Note: When the RXFIFO outflow interrupt is enabled and TXOVIF is detected as 1, an overflow interrupt will be generated</p>	1'b1
[0]	RW	<p>RXUDIMASK</p> <p>RXFIFO underflow interrupt enable bit</p> <p>1'b0: Interrupt disabled</p> <p>1'b1: Interrupt enable</p> <p>Note: When the RXFIFO underflow interrupt is enabled and TXUDIF is detected as 1, an underflow interrupt will be generated</p>	1'b1

16.9.4 Interrupt Flag Register

Table 133 I2S Interrupt Flag Register

bit access		Instructions	reset value
[31:13] RO		RSV	19'h0
[12]	RO	<p>TXIF</p> <p>I2S transmit interrupt flag</p> <p>1'b0: No I2S interrupt occurred</p>	1'b0

		1'b1: I2S has a transmit interrupt generated	
[11]	RO	RXIF I2S receive interrupt flag 1'b0: No I2S interrupt occurred 1'b1: I2S has a receive interrupt generated	1'b0
[10]	RO	I2SIF I2S interrupt flag bit 1'b0: No I2S interrupt occurred 1'b1: I2S has interrupt generation Note: This bit is set whenever either RX or TX has an interrupt	1'b0
[9]	RW	LZCIF Left channel zero-cross detection flag This bit indicates that the left channel next sample data sign bit is changed or all data bits are zero. 1'b0: zero crossing not detected 1'b1: zero crossing detected Note: write 1 to clear the interrupt flag	1'b0
[8]	RW	RZCIF Right channel zero-cross detection flag This bit indicates that the right channel next sample data sign bit has changed or all data bits are zero. 1'b0: zero crossing not detected 1'b1: zero crossing detected Note: write 1 to clear the interrupt flag	1'b0

[7]	RW	<p>TXDONEIF</p> <p>Transmit complete interrupt flag</p> <p>1'b0: This send is not completed</p> <p>1'b1: This transmission is completed</p> <p>Note: write 1 to clear the interrupt flag</p>	1'b0
[6]	RO	<p>TXTHIF</p> <p>RXFIFO interrupt flag</p> <p>1'b0: The number of words in the TXFIFO is greater than the threshold</p> <p>1'b1: The number of words in the TXFIFO is less than or equal to the threshold.</p> <p>Note: When the number of words in the TXFIFO (TxCnt) is equal to or less than the threshold set by TXTH, this bit will change to 1 until data is written to the TXFIFO and the value of TxCnt is greater than the value of TXTH</p> <p>Set to 1 until data is written to the TXFIFO and the value of TxCnt is greater than the value of TXTH</p> <p>change back to 0.</p>	1'b0
[5]	RW	<p>TXOVIF</p> <p>TXFIFO overflow interrupt flag</p> <p>1'b0: TXFIFO overflow interrupt does not occur</p> <p>1'b1: TXFIFO overflow interrupt occurred</p> <p>Note: write 1 to clear the interrupt flag</p>	1'b0
[4]	RW	<p>TXUDIF</p> <p>TXFIFO underflow interrupt flag</p> <p>1'b0: No underflow interrupt occurred in TXFIFO</p> <p>1'b1: TXFIFO underflow interrupt occurred</p> <p>Note: write 1 to clear the interrupt flag</p>	1'b0

[3]	RW	RXDONEIF Receive complete interrupt flag 1'b0: This reception is not completed 1'b1: This reception is completed Note: write 1 to clear the interrupt flag	1'b0
[2]	RO	RXTHIF RXFIFO interrupt flag 1'b0: The number of words in the RXFIFO is less than the threshold 1'b1: The number of words in the RXFIFO is equal to or greater than the threshold. Note: When the number of words in RXFIFO is equal to or more than the threshold set by RXTH, this bit will be set to 1, it will not change back to 0 until the data in the RXFIFO is read and the value of RXCNT is less than the value of RXTH.	1'b0
[1]	RW	RXOVIF RXFIFO overflow interrupt flag 1'b0: RXFIFO overflow interrupt does not occur 1'b1: RXFIFO overflow interrupt occurred Note: write 1 to clear overflow interrupt	1'b0
[0]	RW	RXUDIF RXFIFO underflow interrupt flag 1'b0: RXFIFO underflow interrupt does not occur 1'b1: RXFIFO underflow interrupt occurred Note: write 1 to clear underflow interrupt	1'b0

16.9.5 Status Register

Table 134 I2S Status Register

bit access		Instructions	reset value
[31:10] RO		RSV	22'h0
[9:8]	RO	<p>VALIDBYTE</p> <p>The number of bytes available in the last word.</p> <p>2'b00: After receiving, all bytes in RXFIFO are available</p> <p>2'b01: 1 byte is available in RXFIFO after reception is complete</p> <p>2'b10: 2 bytes are available in the RXFIFO after reception is complete</p> <p>2'b11: 3 bytes are available in the RXFIFO after reception is complete</p>	2'h0
[7:4]	RO	<p>TXCNT</p> <p>Record the number of words in the TXFIFO at the current moment.</p> <p>4'b0000: no data</p> <p>4'b0001: has 1 word</p> <p>...</p> <p>4'b1000: has 8 characters</p>	4'h0
[3:0]	RO	<p>RXCNT</p> <p>Record the number of words in the RXFIFO at the current moment.</p> <p>4'b0000: no data</p> <p>4'b0001: has 1 word</p> <p>...</p> <p>4'b1000: has 8 characters</p>	4'h0

16.9.6 Data Transmission Register

Table 135 I2S data transmission register

bit access		Instructions	reset value
[31:0] WO		<p>TXFIFO</p> <p>The I2S has a built-in 8-word FIFO for storing the data to be sent. Writes one word to the TXFIFO at a time, The word in the TXFIFO is incremented by one. The I2S controller will automatically send out the word that goes into the TXFIFO first.</p>	32'h0

16.9.7 Data Receive Register

Table 136 I2S data receive register

bit access		Instructions	reset value
[31:0] RO		<p>RXFIFO</p> <p>The I2S has a built-in 8-word FIFO for storing the received data. Read one at a time from the RXFIFO word, there will be one less word in the RXFIFO.</p>	32'h0

17 UART module

17.1 Function overview

UART is a universal serial data bus used for asynchronous communication. The bus supports bidirectional communication and can realize full duplex transmission and reception.

W800 has a total of 6 groups of common UART ports. Various baud rate settings can be realized through fine clock frequency division combination, and the maximum support is 2Mbps.

communication rate. The W800 UART can be used in conjunction with hardware DMA to achieve efficient asynchronous transfer of data.

17.2 Main Features

- ÿ Compliant with APB bus interface protocol, full-duplex asynchronous communication mode
- ÿ Support interrupt or polling working mode
- ÿ Support DMA Byte transfer mode, send and receive each 32-byte FIFO
- ÿ Programmable baud rate, maximum support 2Mbps
- ÿ 5-8bit data length, and parity polarity can be configured
- ÿ 1 or 2 stop bits configurable
- ÿ Support RTS/CTS flow control
- ÿ Support Break frame sending and receiving
- ÿ Support Overrun, parity error, frame error, rx break frame interrupt indication

17.3 Functional Description

17.3.1 UART Baud Rate

Asynchronous communication requires both parties to send and receive data according to the negotiated baud rate because the two sides do not have the same clock source for reference. W800

Fine baud rate control can be achieved through the baud rate setting register BAUD_RATE_CTRL register

BAUD_RATE_CTRL[15:0] is named ubdiv, BAUD_RATE_CTRL[19:16] is named ubdiv_frac, the wave to be set

The baudrate, the calculation formula is as follows:

$$\text{ubdiv} = \text{apbclk} / (16 * \text{baudrate}) - 1 \quad // \text{get integer}$$

$$\text{ubdiv_frac} = (\text{apbclk \% (baudrate * 16)}) / \text{baudrate} //\text{Integer}$$

Take the APB clock of 40MHz and the baud rate of 19200bps as an example:

$$\text{ubdiv} = 40000000 / (16 * 19200) - 1 = 129$$

$$\text{ubdiv_frac} = (40000000 \% (19200 * 16)) / 19200 = 3$$

According to the above formula, when the APB clock is 40MHz and the baud rate is 19200bps, the baud rate register should be set to:

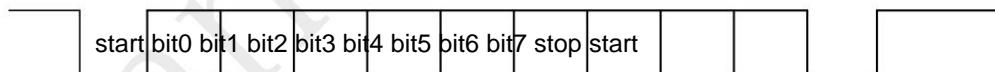
$$\text{BAUD_RATE_CTRL} = (3 << 16) | 129 = 0x0003_0081.$$

17.3.2 UART Data Format

ÿData length

The UART of W800 supports configurable data length of 5bit, 6bit, 7bit and 8bit. The definition of data length is as follows:

8bit single data length



7bit single data length



Figure 28 UART data length

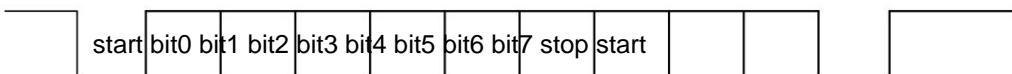
Normal UART communication consists of 1bit start bit, 1bit stop bit plus the middle data bit, and the middle data bit can be configured,

W800 supports 4 data bit lengths of 5bit, 6bit, 7bit, and 8bit that can be configured, and the data bit length can be selected according to the actual application.

ÿ Stop bit

The UART of W800 supports configurable 1bit stop bit and 2bit stop bit, which can be configured according to actual needs, as follows:

1bit stop bit



2bit stop bit

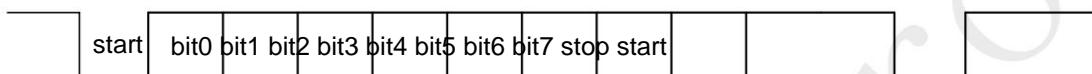


Figure 29 UART stop bit

ÿ Parity bit

The function of the parity bit is to check the correctness of the data, W800 can set odd check, even check and no check.

Odd check calculation method: if the number of current data bits is odd, the odd check bit is 0, if the number of current data bits is even

The odd parity bit is 1. In short, an odd number of 1s is guaranteed.

Calculation method of even parity: if the number of current data bits is odd, the even parity bit is 1, if the number of current data bits is even

Several, the even parity bit is 0. In short, an even number of 1s is guaranteed.

8bit single data length + odd check

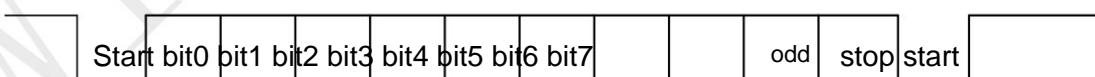
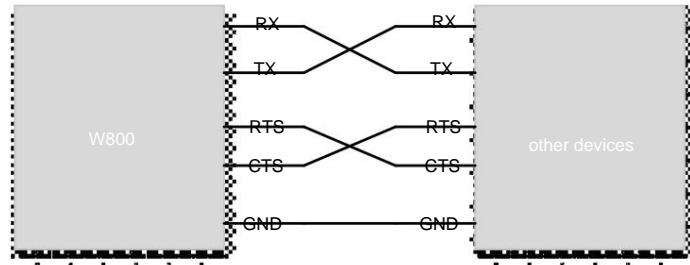


Figure 30 UART parity bit

17.3.3 UART Hardware Flow Control

W800 UART supports hardware flow control in RTS/CTS mode. The main purpose of flow control is to prevent the UART fifo from



The data is lost because the software is too late to process it. RTS and CTS are used correspondingly, as shown in the following figure:

Figure 31 UART hardware flow control connection

The hardware flow control of W800 is controlled by the AUTO_FLOW_CTRL register. When hardware flow control AUTO_FLOW_CTRL[0] is 1

, W800 will set the flow control according to the number of data in rxfifo set by AUTO_FLOW_CTRL[4:2].

When RTS is pulled high, other devices will no longer send data to W800. If the number is less than the set number, RTS will be pulled low, and other devices will continue to send data to W800.

according to. When AUTO_FLOW_CTRL[0] is 0, the software sets the level of RTS through AUTO_FLOW_CTRL[1].

When W800 sends data, it can judge whether the current CTS has changed through interrupt, and query the CTS status through FIFO_STATUS[12],

to decide whether to continue sending data to other devices.

17.3.4 UART DMA transfers

The UART of W800 supports DMA transfer mode. When DMA transfer, you need to configure the DMA_CTRL register in the UART register list.

device to turn on the DMA enable of the UART. At the same time, you need to configure UART_FIFO_CTRL to configure how many bytes are left in txfifo and rxfifo.

Send DMA for transportation.

The source or destination address of DMA is set to TX_DATA_WINDOW or RX_DATA_WINDOW, and the setting parameters of other DMA registers are

See the chapter on DMA registers.

Note: UART DMA transfer can only be set to Byte mode, half_word and word transfer modes are not supported.

17.3.5 UART Interrupts

UART supports interrupt operation mode, including fifo empty, fifo reaching the set trigger value, CTS change, error, etc. will generate UART interrupt,

The desired interrupt can be set through the INT_MASK register.

When the UART interrupt is generated, the current interrupt status can be inquired through INT_SRC, and the reason for the interrupt is triggered. Software writes 1 to clear to 0.

17.4 Register Description

17.4.1 Register List

Table 137 UART register list

offset address	name	abbreviation	access	describe	reset value
0X0000	Data flow control register	UART_LINE_CTRL	Data format setting for RW uart communication		0X0000_000B
0X0004	Automatic hardware flow control register	AUTO_FLOW_CTRL	RW uart rts/cts Hardware flow control setting	0X0000_0014	
0X0008	DMA setup register	DMA_CTRL	RW uart dma transfer mode setting		0X0000_0024
0X000C	FIFO Control Register	UART_FIFO_CTRL	RW set uart fifo trigger level		0X0000_0014
0X0010	Baud rate control register	BAUD_RATE_CTRL	RW Set uart communication baud rate		0X0003_0081
0X0014	Interrupt Mask Register	INT_MASK	RW Set the interrupt 0X0000_01FF that uart needs to use		
0X0018	Interrupt Status Register	INT_SRC	RW uart interrupt status indication		0X0000_0000

0X001C	FIFO Status Register	FIFO_STATUS	RW fifo status, cts status query	0X0000_1000
0X0020	TX start address register TX_DATA_WINDOW WO			0X0000_0000
0X0024 Reserved				
0X0028 Reserved				
0X002C Reserved				
0X0030	RX start address register RX_DATA_WINDOW RO			0X0000_0000
0X0034 Reserved				
0X0038 Reserved				
0X003C Reserved				

17.4.2 Data Flow Control Register

Table 138 UART Data Flow Control Register

bit access		Instructions	reset value
[31:8]		reserve	
[7]	RW	uart_rx_enable Receive enable, active high	1'b0
[6]	RW	uart_tx_enable Send enable, active high.	1'b0
[5]	RW	send break enable Send a break packet. Uart will send a break packet after it is set, and the sending is complete <small>It is automatically cleared to 0 after that.</small>	1'b0
[4]	RW	parity polarity	1'b0

		1'b0: Even parity 1'b1: odd parity	
[3]	RW	parity en Parity check enabled, active high	1'b1
[2]	RW	number of stop bits 1'b0: 1 stop bit 1'b1: 2 stop bits	1'b0
[1 : 0] RW		uart bit length. 2'h0: 5bit 2'h1: 6bit 2'h2: 7bit 2'h3: 8bit	2'h3

17.4.3 Automatic Hardware Flow Control Register

Table 139 UART Auto Hardware Flow Control Register

bit access		Instructions	reset value
[31:5]		reserve	
[4 : 2] RW		RTS trigger level Determines when RTS needs to be deasserted while afc_enable is active. 3'h0: rxfifo has more than 4 bytes 3'h1: rxfifo has more than 8 bytes	3'h5

		3'h2: rx fifo has more than 12 bytes 3'h3: rx fifo has more than 16 bytes 3'h4: rx fifo has more than 20 bytes 3'h5: rx fifo has more than 24 bytes 3'h6: rx fifo has more than 28 bytes 3'h7: rx fifo has more than 31 bytes	
[1]	RW	RTS set When AFC_enable is inactive, software can perform receive flow control by setting this bit. when This bit doesn't care when AFC_enable is active.	1'b0
[0]	RW	afc enable 1'b1: Valid, receive condition rts is generated using rts_trigger_level control.	1'b0

17.4.4 DMA Setup Register

Table 140 UART DMA Setup Register

bit access		Instructions	reset value
[31:8]		reserve	
[7 : 3] RW		rx fifo timeout num When the data in rx fifo is less than rx fifo_trigger_level, if there is no data within N packets When new data is received, an rx fifo timeout interrupt is generated. After the timing function is enabled, whether it is the first timing or the last timing is completed, it will only be Start timing after packets	5'h04
[2]	RW	rx fifo timeout en	1'b1

		rxfifo timeout enable	
[1]	RW	rx dma enable Receive DMA enable, active high. <small>0: Indicates that the receiving process uses interrupts.</small>	1'b0
[0]	RW	tx dma enable Transmit DMA enable, active high. <small>0: Indicates that the sending process uses interrupts.</small>	1'b0

17.4.5 FIFO Control Register

Table 141 UART FIFO Control Register

bit access		Instructions	reset value
[31:6]		reserve	
[5 : 4] RW		rxfifo trigger level <small>When the number of data bytes in rxfifo is greater than or equal to this value, an interrupt is triggered, or rxdma req is triggered.</small> 2'h0: 1byte 2'h1: 4byte 2'h2: 8byte 2'h3: 16byte	2'h1
[3 : 2] RW		txfifo trigger level <small>When the number of data bytes in txfifo is less than or equal to this value, an interrupt is triggered, or txdma req is triggered.</small> 2'h0: empty 2'h1: 4byte	2'h1

		2'h2: 8byte 2'h3: 16byte	
[1]	RW	rxfifo reset Reset rxfifo, clear rxfifo state	1'b0
[0]	RW	txfifo reset Reset txfifo, clear txfifo state	1'b0

17.4.6 Baud Rate Control Register

Table 142 UART Baud Rate Control Register

bit access		Instructions	reset value
[31:20]		reserve	
[19:16] RW		ubdiv_frac Indicates the fractional part of the system clock divided by 16 times the baud rate clock quotient. The specific value is fracx16. (Refer to Section 2.3.2, Baud Rate Calculation Method)	4'h3
[15:0] RW		ubdiv The integer part of the system clock divided by 16 times the baud rate clock quotient minus one. The default system clock is 40MHz and the baud rate is 19200. (Refer to Section 2.3.2, Baud Rate Calculation Method)	16'h81

17.4.7 Interrupt Mask Register

Table 143 UART Interrupt Mask Register

bit access		Instructions	reset value

[31:9]		reserve	
[8]	RW	overrun error int mask, rxfifo overflow interrupt mask bit, active high.	1'b1
[7]	RW	parity error int mask, parity check interrupt mask bit, active high.	1'b1
[6]	RW	frame error int mask, data frame error interrupt mask bit, active high.	1'b1
[5]	RW	break detect int mask, break signal detection interrupt mask bit, active high.	1'b1
[4]	RW	cts changed indicate mask, CTS signal change interrupt mask bit, active high.	1'b1
[3]	RW	rxfifo data timeout int mask, rxfifo receive data timeout interrupt mask bit, active high.	1'b1
[2]	RW	rxfifo trigger level int mask, rxfifo reaches the trigger value interrupt mask bit, active high.	1'b1
[1]	RW	txfifo trigger level int mask, txfifo reaches the trigger value interrupt mask bit, active high.	1'b1
[0]	RW	txfifo empty int mask, txfifo is the empty interrupt mask bit, active high.	1'b1

17.4.8 Interrupt Status Register

Table 144 UART Interrupt Status Register

bit access		Instructions	reset value
[31:9]		reserve	
[8]	RW	overrun error rxfifo overflowed. <small>Software actively writes 1 to clear to 0.</small>	1'b0
[7]	RW	parity error <small>The received packet has an incorrect parity bit.</small> <small>In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt.</small> <small>Software actively writes 1 to clear to 0.</small>	1'b0

[6]	RW	<p>frame error</p> <p>Received packet with stop bit error.</p> <p>In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[5]	RW	<p>break detect</p> <p>A break packet is received.</p> <p>In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[4]	RW	<p>cts changed</p> <p>This interrupt is generated when the cts signal changes.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[3]	RW	<p>rxfifo data timeout</p> <p>The data length in rx fifo is less than rx fifo trigger level but no data is received for N data cycles,</p> <p>An interrupt is generated.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[2]	RW	<p>rxfifo trigger level interrupt</p> <p>When the number of data in rx fifo changes from less than the number specified in rx fifo trigger level to greater than or equal to the number , this interrupt is generated.</p> <p>At this point, the current data frame size should be determined according to the rx fifo count.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[1]	RW	<p>txfifo trigger level interrupt</p> <p>When the number of data in tx fifo changes from greater than the number specified in tx fifo trigger level to less than or equal to the number</p>	1'b0

		, an interrupt is generated. Software actively writes 1 to clear to 0.	
[0]	RW	tx fifo empty interrupt This interrupt is generated when the current packet is sent and the txfifo is empty. Software actively writes 1 to clear to 0.	1'b0

17.4.9 FIFO Status Register

Table 145 UART FIFO Status Register

bit access		Instructions	reset value
[31:13]		reserve	
[12]	RW	cts status the state of the current cts	1'b0
[11:6] RW		rxfifo count Number of data in rxfifo	6'h0
[5 : 0] RW		txfifo count Number of data in txfifo	6'h0

17.4.10 TX Start Address Register

Table 146 UART TX Start Address Register

bit access		Instructions	reset value
[31:0] WO		tx data window Send data start address.	32'h0

		<p>Note: UART only supports byte operation for sending and receiving data. When using burst transmission, it is possible to use word</p> <p>The section address is incremented, and the design supports up to 16-burst operations, that is, 16byte. So from sending /</p> <p>After receiving the starting address, a total of 16bytes (4 words) are reserved as the send/receive data window.</p>	
--	--	---	--

17.4.11RX Start Address Register

Table 147 UART RX Start Address Register

bit access		Instructions	reset value
[31:0] RO		<p>rx data window</p> <p>Receive data start address.</p> <p>Note: UART only supports byte operation for sending and receiving data. When using burst transmission, it is possible to use word</p> <p>The section address is incremented, and the design supports up to 16-burst operations, that is, 16byte. So from sending /</p> <p>After receiving the starting address, a total of 16bytes (4 words) are reserved as the send/receive data window.</p>	32'h0

18 UART&7816 module

18.1 Function overview

UART&7816 module is compatible with UART function and 7816 interface function.

W800 supports 1 set of UART&7816 multiplexing interface (uart2), when used as UART, it can be

Various baud rate settings can be realized, and the maximum communication rate of 2Mbps can be supported.

The W800 UART&7816 interface can be used in conjunction with hardware DMA to achieve efficient data transmission.

18.2 Main Features

ÿ Compliant with the APB bus interface protocol, supporting UART asynchronous full-duplex and 7816 asynchronous half-duplex communication modes;

ÿ Support interrupt or polling working mode;

ÿ Support DMA Byte transmission mode, send and receive each 32-byte FIFO;

ÿ Serial port function:

ÿ Programmable baud rate, maximum support 2Mbps

ÿ 5-8bit data length, and parity polarity can be configured

ÿ 1 or 2 stop bits configurable

ÿ Support RTS/CTS flow control

ÿ Support Break frame sending and receiving

ÿ Support Overrun, parity error, frame error, rx break frame interrupt indication

ÿ 7816 interface function:

ÿ Compatible with ISO-7816-3 T=0.T=1 mode

- ÿ Compatible with EVM2000 protocol
- ÿ Configurable guard time (11 ETU-267 ETU)
- ÿ Forward/reverse convention, software configurable
- ÿ Support send/receive parity check and retransmission function
- ÿ Support 0.5 and 1.5 stop bit configurable

18.3 UART function description

Refer to Chapter 16 UART Function Module Description

18.4 7816 Functional Description

18.4.1 Introduction to 7816

ISO7816 is an international standard smart card protocol, which specifies the physical characteristics, size and interface, electrical signal and transmission protocol, life order, safety and other information.

W800 mainly implements the part of ISO 7816-3 electrical signal and transmission protocol, and supports T0 and T1 cards. Via 7816 of W800

The user can not care about the signal communication logic of the clock and data, and can directly interact with the smart card for data and commands. About Smart

The user needs to refer to the ISO7816-4 protocol to realize the data and command interaction mode of the energy card.

18.4.2 7816 interface

W800 mainly integrates two interfaces of smart card clock and data to realize the communication electrical signal logic of data and command. The actual smart card should

In use, there are three signals of RST, VCC and GND. RST can be controlled by ordinary GPIO, mainly when the smart card is powered on and reset.

use. VCC can be directly connected to the 3.3V power supply, or through GPIO with other circuits to control the on-off of the VCC of the smart card.

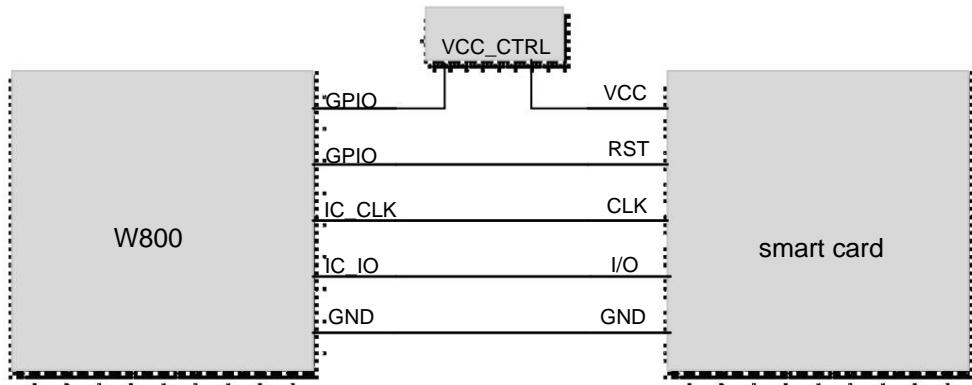


Figure 32 7816 Connection Diagram

18.4.3 7816 Configuration

When used as a 7816 interface, related configuration is required:

ÿ Set the interface working mode, UART_LIN_CTRL[24] is set to 1, and the current interface is selected as 7816 mode;

ÿ Set 7816 MSB or LSB transmission mode, UART_LIN_CTRL[3] is set to 1, through UART_LIN_CTRL[3]

Select whether the 7816 interface is in MSB mode (bit7 is transmitted first) or LSB mode (bit0 is transmitted first);

ÿ Set stop bits, UART_LIN_CTRL[2] can select 0.5 or 1.5 stop bits for smart card;

ÿ Select card type, UART_LIN_CTRL[8] can select T0 card or T1 card;

ÿ Configure the smart card communication timeout time, set the timeout time through WAIT_TIME, when receiving data, if the data is not received after the timeout

A timeout interrupt is generated.

18.4.4 7816 Clock Configuration

The smart card clock refers to the clock provided to the smart card through the CLK pin, set by BAUD_RATE_CTRL[21:16], calculate

Methods as below:

$$\text{clk_div} = \frac{-}{2 \times -} \circlearrowleft 1$$

fsc_clk: CLK that needs to be provided to the smart card;

fclk_apb: system APB bus clock;

clk_div: The clock division factor that needs to be set to BAUD_RATE_CTRL[21:16]

Because clk_div can only take integers, in order to reduce errors, we'd better adopt the rounding calculation method, the rounding of C language calculation will be lost.

Drop the decimal part, the C language adopts the rounding conversion method as follows:

$$\text{clk_div} = (\text{fclk_apb} + \text{fsc_clk}) / (2 * \text{fsc_clk}) - 1;$$

18.4.5 7816 Rate Settings

There is a time unit ETU in the smart card, and the smart card transmits data and commands according to this time unit. The ETU is set by

BAUD_RATE_CTRL[15: 0] to set, the calculation method is as follows:

$$1 \text{ etu} = \frac{x}{f}$$

f: the CLK of our smart card;

Both F and D are parameters given by the smart card.

In fact, the ubdiv of BAUD_RATE_CTRL[15: 0] we need to set is F/D. The above formula is only for calculating ETU for large

home reference. When we actually set it, we only need to set ubdiv = F/D. F and D can be queried from the table below.

Table 7 — F_i and $f_{\max.}$

Bits 8 to 5	0000	0001	0010	0011	0100	0101	0110	0111
F_i	372	372	558	744	1116	1488	1860	RFU
$f_{\max.}$ MHz	4	5	6	8	12	16	20	—
Bits 8 to 5	1000	1001	1010	1011	1100	1101	1110	1111
F_i	RFU	512	768	1024	1536	2048	RFU	RFU
$f_{\max.}$ MHz	—	5	7.5	10	15	20	—	—

— According to Table 8, bits 4 to 1 encode D_i .

Table 8 — D_i

Bits 4 to 1	0000	0001	0010	0011	0100	0101	0110	0111
D_i	RFU	1	2	4	8	16	32	64
Bits 4 to 1	1000	1001	1010	1011	1100	1101	1110	1111
D_i	12	20	RFU	RFU	RFU	RFU	RFU	RFU

Table 148 7816 Rate Settings

(Refer to the protocol document ISO_IEC_FDIS_7816-3_(E).PDF)

18.4.6 7816 Power-on Reset

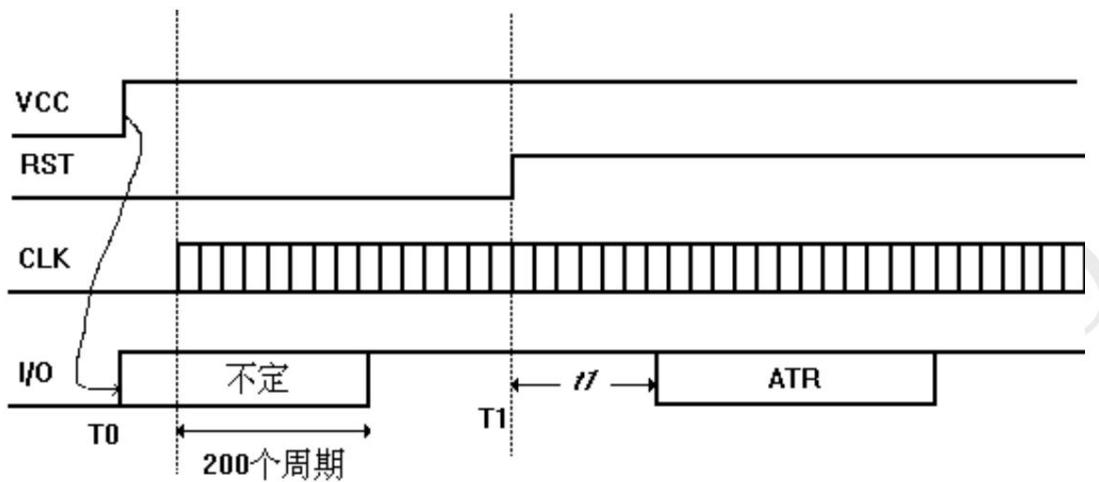


Figure 33 7816 power-on reset sequence

The figure above is the timing diagram of the power-on reset of the smart card. The initial state of CLK and IO is low, we need to configure it into GPIO mode and pull it low. VCC pull

After high, CLK and IO can be controlled by 7816 after being configured in 7816 mode. Finally, we need to manually pull the RST pin high to complete the reset

Procedure. The configuration steps are as follows:

ÿ I/O, CLK, RST are configured as normal GPIO mode and keep low level;

ÿ Set 7816 to $T=0$ mode;

ÿ Control VCC power-on through GPIO;

ÿ Configure I/O and CLK as 7816 mode, and 7816 drives clock and data;

ÿ Configure 7816 clock frequency and enable clock output;

ÿ Set the RST pin and wait for the ATR data to be received. If the ATR data is not received within 40000 clocks, the deactivation process will be executed.

Card is deactivated.

18.4.7 7816 Warm reset

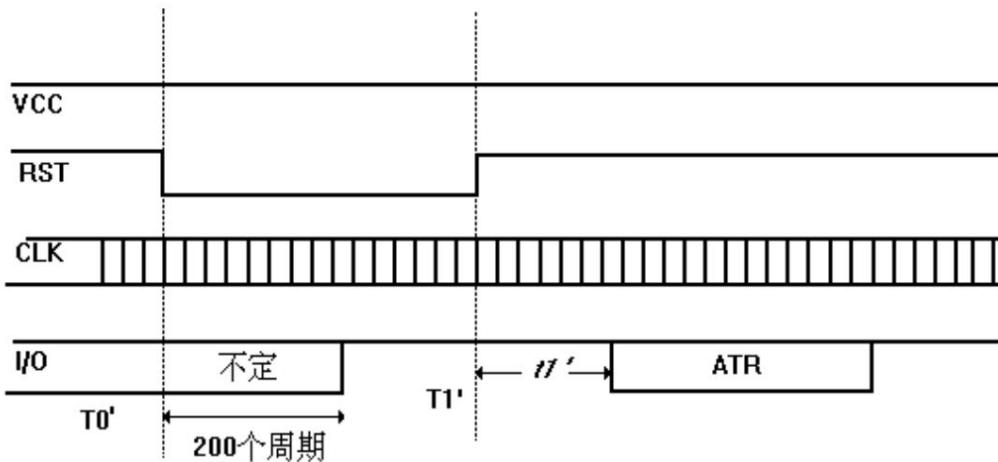


Figure 34 7816 warm reset

As shown in the figure above, the process of warm reset is very simple. In normal working mode, the RST pin can be pulled down for 400 cycles. The configuration steps are as follows:

ÿ Keep VCC powered on;

ÿ Pull down the RST pin for at least 400 clock cycles;

ÿ Pull the RST pin high and wait for the ATR data to be received. If the ATR data is not received within 40000 clocks, the deactivation process will be executed.

Card is deactivated.

18.4.8 7816 Inactivation process

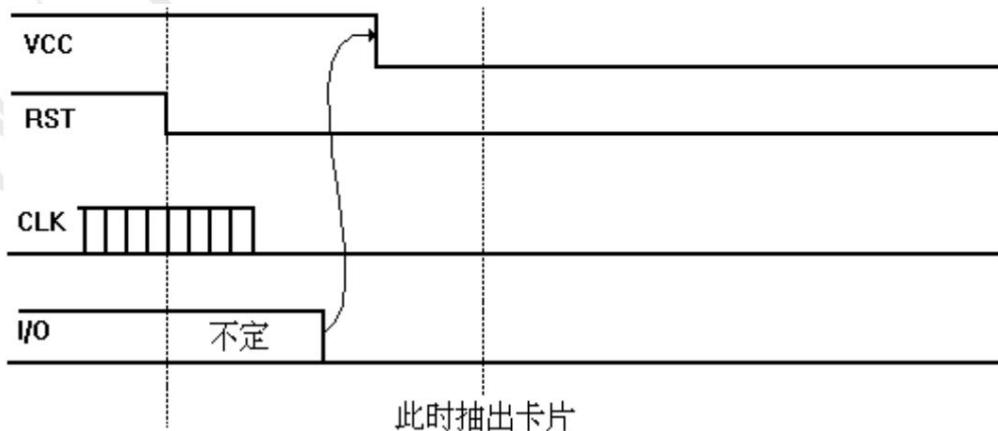


Figure 35 7816 inactivation process

As shown in the figure above, after RST is pulled low, it is necessary to configure CLK and IO into normal IO mode and pull it down, and finally turn off the VCC power supply, the operation steps

as follows:

ÿ Keep VCC powered on;

ÿ Pull down the RST pin;

ÿ Configure CLK and IO as GPIO mode and pull them low;

ÿ Control VCC power down through GPIO;

18.4.9 7816 Data Transfer

The data transmission sequence of 7816 has been completed by W800 hardware, and no user operation is required. If the user wants to know the specific content of this part, please

Refer to the provisions in the ISO7816-3 protocol.

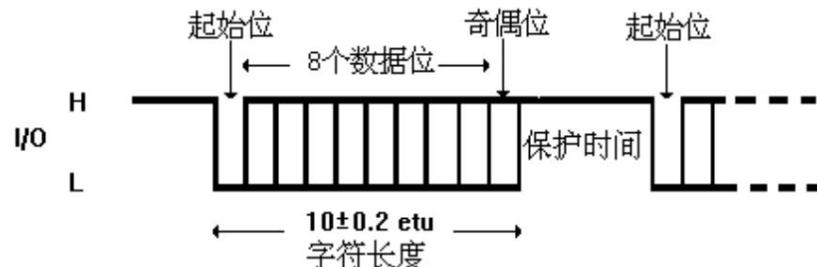


Figure 36 7816 data transmission

18.4.10UART&7816 DMA transfer

The UART&7816 of W800 supports DMA transfer mode, and the UART&7816 register list needs to be configured during DMA transfer.

DMA_CTRL register to enable DMA for UART. At the same time, you need to configure UART_FIFO_CTRL in txfifo and rxfifo

How many bytes are left to trigger DMA transfers.

The source or destination address of DMA is set to TX_DATA_WINDOW or RX_DATA_WIDNOW, and the setting parameters of other DMA registers are

See the chapter on DMA registers.

Note: UART&7816 DMA transfer can only be set to Byte mode, half_word and word transfer modes are not supported.

18.4.11 UART&7816 Interrupt

UART&7816 supports interrupt operation mode, including fifo empty, fifo reaching the set trigger value, CTS changes, errors, etc. will be generated

UART&7816 interrupt, the required interrupt can be set through the INT_MASK register.

When the UART&7816 interrupt is generated, the current interrupt status can be inquired through INT_SRC, and the reason for the interrupt is triggered. Software write 1 clear 0.

18.5 Register Description

18.5.1 Register List

Table 149 UART&7816 register list

offset address	name	abbreviation	access	describe	reset value
0X0000 Data flow control register		UART_LINE_CTRL	Data related equipment for RW uart&7816 communication set		0X0033_520B
0X0004 Automatic hardware flow control register AUTO_FLOW_CTRL	RW uart rts/cts		Hardware flow control setting 0X0000_0014		
0X0008 DMA setup register	DMA_CTRL	RW	uart&7816 dma transfer mode setting 0X0000_0024		
0X000C FIFO Control Register	UART_FIFO_CTRL	RW	set uart&7816 fifo trigger level 0X0000_0014		
0X0010 Baud rate control register	BAUD_RATE_CTRL	RW	set uart baud rate, 7816 clock 0X0003_0082		
0X0014 Interrupt Mask Register	INT_MASK	RW	set uart&7816 need to use broken		0X0000_03FF

0X0018	Interrupt Status Register	INT_SRC	RW uart&7816 interrupt status indication	0X0000_0000
0X001C	FIFO Status Register	FIFO_STATUS	RW fifo status, cts status query	0X0000_0000
0X0020	TX start address register TX_DATA_WINDOW WO			0X0000_0000
0X0024	Reserved			
0X0028	Reserved			
0X002C	Reserved			
0X0030	RX start address register RX_DATA_WINDOW RO			0X0000_0000
0X0034	Reserved			
0X0038	Reserved			
0X003C	Reserved			
0X0040	7816 Guard time register GUARD_TIME	RW 7816	Guard time between data	0X0000_0000
0X0044	7816 Timeout time register WAIT_TIME	RW 7816	Receive data timeout time	0X0007_8000

18.5.2 Data Flow Control Register

Table 150 UART&7816 data flow control register

bit access		Instructions	reset value
[31:25]		reserve	
[twenty four]	RW	sc_mode 1'b0: uart mode 1'b1: 7816 mode	1'b0
[twenty three]	RW	7816 card T0 mode rx_retrans_en 1'b0: Rx automatic retransmission is invalid	1'b0

		1'b1: Rx automatic retransmission enable	
[22:20] RW		7816 card T0 mode rx_retrans_cnt	3'h3
[19]	RW	<p>7816 card T0 mode tx_retrans_en</p> <p>1'b0: Tx automatic retransmission is invalid</p> <p>1'b1: Tx automatic retransmission enable</p>	1'b0
[18:16] RW		<p>7816 card T0 mode tx_retrans_cnt</p> <p>tx automatic retransmission times</p>	3'h3
[15:11] RW		<p>The minimum MIN_BGT (Min Block Guard Time) of the 7816 card</p> <p>Min Block Guard Time calculation: 10+stop bits (default 2 bits)+configured value MIN_BGT</p> <p>Note:</p> <p>T=0: The minimum time interval between the falling edges of the start bits of consecutive characters in opposite directions of transmission and reception</p> <p>Cannot be less than 16 ETUs. Must be able to correctly interpret the received falling edge of its start bit and the last word transmitted</p> <p>The interval between the falling edges of the section start bit is 15 ETU characters.</p> <p>T=1: The minimum time interval between the falling edges of the start bits of consecutive characters in opposite directions of transmission and reception</p> <p>(Block Guard Time, BGT) must be 22 ETUs. must be able to interpret the received start bit correctly</p> <p>The interval between the falling edge and the falling edge of the last sent byte start bit is the character received within 21 ETUs.</p>	5'ha
[10]	RW	<p>7816 Card Clock Control Configuration</p> <p>1'b0: The card clock output is generated when configured in card mode, otherwise the card clock output is invalid</p> <p>1'b1: Clock stopped</p>	1'b0
[9]	RW	<p>Whether to receive data when the parity of the 7816 card is wrong</p> <p>1'b0: do not receive</p> <p>1'b1: receive</p>	1'b1

[8]	RW	7816 card T0/T1 mode configuration, 1'b0: T0 mode 1'b1: T1 mode	1'b0
[7]	RW	uart_rx_enable In uart/7816 mode, receive enable, active high	1'b0
[6]	RW	uart_tx_enable In uart/7816 mode, transmit enable, active high.	1'b0
[5]	RW	send break enable Send a break packet. Uart will send a break packet after it is set, and the sending is complete <small>It is automatically cleared to 0 after that.</small>	1'b0
[4]	RW	parity polarity (UART mode) 1'b0: Even parity 1'b1: odd parity Forward and reverse (7816 mode) 1'b0: LSB (b0 bit) is transmitted first 1'b1: MSB (b7 bit) is transmitted first	1'b0
[3]	RW	parity enable, active high (UART mode)	1'b1
[2]	RW	Number of stop bits (UART mode) 1'b0: 1 stop bit 1'b1: 2 stop bits Number of stop bits (7816 mode) 1'b0: 0.5 stop bits	1'b0

		1'b1: 1.5 stop bits	
[1 : 0] RW		uart bit length (UART mode) 2'h0: 5bit 2'h1: 6bit 2'h2: 7bit 2'h3: 8bit	2'h3

18.5.3 Automatic Hardware Flow Control Register

Table 151 UART&7816 Automatic Hardware Flow Control Register

bit access		Instructions	reset value
[31:5]		reserve	
[4 : 2] RW		RTS trigger level (UART mode) Determines when RTS needs to be deasserted while afc_enable is active. 3'h0: rxfifo has more than 4 bytes 3'h1: rxfifo has more than 8 bytes 3'h2: rxfifo has more than 12 bytes 3'h3: rxfifo has more than 16 bytes 3'h4: rxfifo has more than 20 bytes 3'h5: rxfifo has more than 24 bytes 3'h6: rxfifo has more than 28 bytes 3'h7: rxfifo has more than 31 bytes	3'h5

[1]	RW	RTS set (UART mode) When AFC_enable is inactive, software can perform receive flow control by setting this bit. when This bit doesn't care when AFC_enable is active.	1'b0
[0]	RW	afc enable (UART mode) Receive condition rts is generated using rts_trigger_level control, high effective.	1'b0

18.5.4 DMA Setup Register

Table 152 UART&7816 DMA setting register

bit access		Instructions	reset value
[31:8]		reserve	
[7 : 3] RW		rx fifo timeout num (UART mode) When the data in rx fifo is less than rx fifo_trigger_level, if there is no data within N packets When new data is received, an rx fifo timeout interrupt is generated. After the timing function is enabled, whether it is the first timing or the last timing is completed, it will only be Start timing after packets	5'h4
[2]	RW	rx fifo timeout en (UART&7816 mode) rx fifo timeout enable, active high	1'b1
[1]	RW	rx dma enable (UART&7816 mode) Receive DMA enable, active high. 0 means the receive process uses interrupts.	1'b0
[0]	RW	tx dma enable (UART&7816 mode) Transmit DMA enable, active high.	1'b0

		0 means that the transmit process uses interrupts.	
--	--	--	--

18.5.5 FIFO Control Register

Table 153 UART&7816 FIFO Control Register

bit access		Instructions	reset value
[31:6]		reserve	
[5 : 4] RW		<p>rxfifo trigger level (UART&7816 mode)</p> <p>When the number of data bytes in rx fifo is greater than or equal to this value, an interrupt is triggered, or rxdma req is triggered.</p> <p>2'h0: 1byte 2'h1: 4byte 2'h2: 8byte 2'h3: 16byte</p>	2'h1
[3 : 2] RW		<p>txfifo trigger level(UART&7816 mode)</p> <p>When the number of data bytes in tx fifo is less than or equal to this value, an interrupt is triggered, or txdma req is triggered.</p> <p>2'h0: empty 2'h1: 4byte 2'h2: 8byte 2'h3: 16byte</p>	2'h1
[1]	RW	<p>rxfifo reset (UART&7816 mode)</p> <p>Reset rx fifo, clear rx fifo state</p>	1'b0
[0]	RW	<p>txfifo reset (UART&7816 mode)</p> <p>Reset tx fifo, clear tx fifo state</p>	1'b0

18.5.6 Baud Rate Control Register

Table 154 UART&7816 Baud Rate Control Register

bit access		Instructions	reset value
[31:20]		reserve	
[19:16] RW		<p>ubdiv_frac</p> <p>UART mode:</p> <p>Indicates the fractional part of the system clock divided by 16 times the baud rate clock quotient. The specific value is fracx16.</p> <p>(Refer to chapter Baud rate calculation method)</p> <p>7816 mode:</p> <p>$ubdiv_frac = (fclk_apb + fsc_clk)/(2 * fsc_clk) - 1;$</p> <p>(Refer to 7816 clock calculation method)</p>	4'h3
[15:0] RW		<p>ubdiv</p> <p>UART mode:</p> <p>The integer part of the system clock divided by 16 times the baud rate clock quotient minus one.</p> <p>The default system clock is 40MHz and the baud rate is 19200.</p> <p>(Refer to the baud rate calculation method)</p> <p>7816 mode:</p> <p>$ubdiv=Fi/Di$ (Fi, Di are the parameters fed back by the smart card, edu frequency: $f_etuckl = fsc_clk/(ubdiv+1)$)</p> <p>(Refer to Section 7816 Rate Calculation Method)</p>	16'h82

18.5.7 Interrupt Mask Register

Table 155 UART&7816 Interrupt Mask Register

bit access		Instructions	reset value
[31:10]		reserve	
[9]	RW	The 7816 card received error signal when sending. (7816 mode)	1'b1
[8]	RW	overrun error int mask, rxfifo overflow interrupt mask bit, active high. (UART&7816 mode)	1'b1
[7]	RW	parity error int mask, parity check interrupt mask bit, active high. (UART&7816 mode)	1'b1
[6]	RW	frame error int mask, data frame error interrupt mask bit, active high. (UART mode)	1'b1
[5]	RW	break detect int mask, break signal detection interrupt mask bit, active high. (UART mode)	1'b1
[4]	RW	cts changed indicate mask, CTS signal change interrupt mask bit, active high. (UART mode)	1'b1
[3]	RW	rxfifo data timeout int mask, rxfifo receive data timeout interrupt mask bit, active high. (UART&7816 mode)	1'b1
[2]	RW	rxfifo trigger level int mask, rxfifo reaches the trigger value interrupt mask bit, active high. (UART&7816 mode)	1'b1
[1]	RW	txfifo trigger level int mask, txfifo reaches the trigger value interrupt mask bit, active high. (UART&7816 mode)	1'b1
[0]	RW	txfifo empty int mask, txfifo is the empty interrupt mask bit, active high. (UART&7816 mode)	1'b1

18.5.8 Interrupt Status Register

Table 156 UART&7816 Interrupt Status Register

bit access		Instructions	reset value
[31:9]		reserve	

[9]	RW	The 7816 card received error signal when sending. (7816 mode)	
[8]	RW	<p>overrun error (UART&7816 mode)</p> <p>rxfifo overflowed.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[7]	RW	<p>parity error (UART&7816 mode)</p> <p>The received packet has an incorrect parity bit.</p> <p>In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[6]	RW	<p>frame error (UART mode)</p> <p>Received packet with stop bit error.</p> <p>In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[5]	RW	<p>break detect (UART mode)</p> <p>A break packet is received.</p> <p>In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[4]	RW	<p>cts changed (UART mode)</p> <p>This interrupt is generated when the cts signal changes.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[3]	RW	<p>rxfifo data timeout (UART&7816 mode)</p> <p>The data length in rxfifo is less than rxfifo trigger level but no data is received for N data cycles,</p> <p>An interrupt is generated.</p>	1'b0

		Software actively writes 1 to clear to 0.	
[2]	RW	<p>rxfifo trigger level interrupt (UART&7816 mode)</p> <p>When the number of data in rxifo changes from less than the number specified in rxifo trigger level to greater than or equal to the number , this interrupt is generated.</p> <p>At this point, the current data frame size should be determined according to the rxifo count.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[1]	RW	<p>txfifo trigger level interrupt (UART&7816 mode)</p> <p>When the number of data in txfifo changes from greater than the number specified in txfifo trigger level to less than or equal to the number , an interrupt is generated.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0
[0]	RW	<p>tx fifo empty interrupt (UART&7816 mode)</p> <p>This interrupt is generated when the current packet is sent and the txfifo is empty.</p> <p>Software actively writes 1 to clear to 0.</p>	1'b0

18.5.9 FIFO Status Register

Table 157 UART&7816 FIFO Status Register

bit access		Instructions	reset value
[31:13]		reserve	
[12]	RW	<p>cts status (UART mode)</p> <p>the state of the current cts</p>	1'b0

[11:6] RW		rxfifo count (UART&7816 mode) Number of data in rxfifo	6'h0
[5 : 0] RW		txfifo count (UART&7816 mode) Number of data in txfifo	6'h0

18.5.10 TX Start Address Register

Table 158 UART&7816 TX start address register

bit access		Instructions	reset value
[31:0] WO		tx data window (UART&7816 mode) Send data start address. Note: UART only supports byte operation for sending and receiving data. When using burst transmission, it is possible to use word The section address is incremented, and the design supports up to 16-burst operations, that is, 16byte. So from sending / After receiving the starting address, a total of 16bytes (4 words) are reserved as the send/receive data window.	32'h0

18.5.11 RX Start Address Register

Table 159 UART&7816 RX start address register

bit access		Instructions	reset value
[31:0] RO		rx data window (UART&7816 mode) Receive data start address. Note: UART only supports byte operation for sending and receiving data. When using burst transmission, it is possible to use word The section address is incremented, and the design supports up to 16-burst operations, that is, 16byte. So from sending / After receiving the starting address, a total of 16bytes (4 words) are reserved as the send/receive data window.	32'h0

18.5.12 7816 Guard Time Register

Table 160 7816 Guard Time Register

bit access		Instructions	reset value
[31:8]		reserve	
[7 : 0] RW		ex_gt_num In 7816 mode, guard time calculation: 10+stop bit+config value ex_gt_num	8'h0

18.5.13 7816 Timeout time register

Table 161 7816 Timeout Time Register

bit access		Instructions	reset value
[31:24]		reserve	
[23:0] RW		wait time counter (in ETU) In 7816 mode: CWT and BWT times, configured as maximum defaults. (In T1 mode: BWT = (11 etu+ 2BWI*960*Fd/fsc))	24'h78000

19 Timer module

19.1 Functional overview

The timer contains a 32-bit auto-loaded counter driven by a divided system clock. W800 has 6 fully independent

timer. Accurate timing and interrupt functions are implemented, which can be used for delayed or periodic event processing.

19.2 Main Features

ÿ 6 fully independent timers

ÿ 32-bit autoload counter

ÿ The timing unit can be configured as ms, us

ÿ It can realize single timing or repeated timing function

ÿ Scheduled interrupt function

ÿ You can query the timer count value at any time;

19.3 Functional Description

The timer module consists of 6 completely independent timers, which do not affect each other, and the 6 channels can work at the same time.

After the system clock is divided by the frequency division factor, the us standard clock is obtained, which is used for the input clock of the counter. Timing unit can be configured as us, ms two kinds of levels.

The timing value is a 32-bit configurable register that can meet the needs of different timing durations. Each timer corresponds to an interrupt, when

After the timing time is met, if the interrupt function is enabled, an interrupt request will be generated, which can be used to process periodic events.

19.3.1 Timing function

Timing function means that according to the time set by the user, when the time is up, a hardware interrupt is generated to notify the user to implement a specific function. Timing trigger support ticket

There are two types, time and period, one can be used to process single events, and the other can be used to process periodic events.

The user obtains the APB bus clock frequency according to the frequency division factor of the system clock, and sets the base microsecond count configuration register of the timer

(TMR_CONFIG), set the timing value, configure the timing unit, work mode, enable the interrupt, and then start the timing function. When timed

When the time is up, the program enters the timer interrupt processing function and clears the interrupt.

19.3.2 Delay function

The delay function means that the user can make the program in a waiting state according to the countdown function of the timer, and the program will continue to run until the timing is completed.

Row.

19.4 Register Description

19.4.1 Register List

Table 162 Timer register list

offset address name		Abbreviated access		describe	reset value
0X0000 Standard us configuration register		TMR_CONFIG RW		Standard us timing divider value, by bus time Divide the frequency to get the standard us timing, this value Equal to APB bus frequency (MHz) minus --	0X0000_0027
0X0004 Timer Control Register		TMR_CSR	RW Timer	Control Register	0X0631_8C63
0X0008 Timer 1 timing value configuration register TMR1_PRD			RW Timer1	timing value configuration register 0X0000_0000	
0X000C Timer 2 timing value configuration register TMR2_PRD			RW Timer2	timing value configuration register 0X0000_0000	
0X0010 Timer 3 timing value configuration register TMR3_PRD			RW Timer3	timing value configuration register 0X0000_0000	

0X0014 Timer 4 timing value configuration register	TMR4_PRD	RW	Timer4 timing value configuration register	0X0000_0000
0X0018 Timer 5 timing value configuration register	TMR5_PRD	RW	Timer5 timing value configuration register	0X0000_0000
0X001C Timer 6 timing value configuration register	TMR6_PRD	RW	Timer6 timing value configuration register	0X0000_0000
0X0020 Timer 1 current count value	TMR1_CNT	RO	Timer1 current count value	0X0000_0000
0X0024 Timer 2 current count value	TMR1_CNT	RO	Timer2 current count value	0X0000_0000
0X0028 Timer 3 current count value	TMR1_CNT	RO	Timer3 current count value	0X0000_0000
0X002C Timer 4 current count value	TMR1_CNT	RO	Timer4 current count value	0X0000_0000
0X0030 Timer 5 current count value	TMR1_CNT	RO	Timer5 current count value	0X0000_0000
0X0034 Timer 6 current count value	TMR1_CNT	RO	Timer6 current count value	0X0000_0000

19.4.2 Standard us configuration registers

Table 163 Timer standard us configuration register

bit access		Instructions	reset value
[31:7]		reserve	
[6 : 0] RW		<p>The clock divider configures prescale. E.g: $apb_clk=40MHz$ $prescale = 40 - 1 = 8'd39$</p>	7'h27

19.4.3 Timer Control Register

Table 164 Timer Timer Control Register

bit access		Instructions	reset value

[31:30] RW		reserve	2'h0
[29:25] RW		TMR6_CSR, same as TMR1_CSR	5'h3
[24:20] RW		TMR5_CSR, same as TMR1_CSR	5'h3
[19:15] RW		TMR4_CSR, same as TMR1_CSR	5'h3
[14:10] RW		TMR3_CSR, same as TMR1_CSR	5'h3
[9:5] RW		TMR2_CSR, same as TMR1_CSR	5'h3
[4 : 0] RW		[4:0] is TMR1_CSR, as follows:	
		[4]: Interrupt status register, write 1 to clear	
		1'b0: Timer generates no interrupt;	1'b0
		1'b1: Timer generates an interrupt;	
		[3]: Interrupt Enable Register	
		1'b0: No interrupt will be generated after the timing is completed; 1'b1: An interrupt is generated after the timing is completed;	1'b0
		[2]: Timer enable register	
		1'b0: the timer does not work; 1'b1: enable timer	1'b0
		[1]: Timer working mode	
		1'b0: Timer repeats timing; 1'b1: The timer is only timed once, and it will automatically close after the time is completed;	1'b1
		[0]: Timer timing unit	
		1'b0: The timing unit is us; 1'b1: The timing unit is ms;	1'b1

19.4.4 Timer 1 Timing Value Configuration Register

Table 165 Timer 1 Timing Value Configuration Register

bit access		Instructions	reset value
[31: 0] RW		Configure the timer value of timer 1	32'b0

19.4.5 Timer 2 Timing Value Configuration Register

Table 166 Timer 2 Timing Value Configuration Register

bit access		Instructions	reset value
[31: 0] RW		Configure the timer value of timer 2	32'b0

19.4.6 Timer 3 Timing Value Configuration Register

Table 167 Timer 3 Timing Value Configuration Register

bit access		Instructions	reset value
[31: 0] RW		Configure the timer value of timer 3	32'b0

19.4.7 Timer 4 Timer Value Configuration Register

Table 168 Timer 4 Timing Value Configuration Register

bit access		Instructions	reset value
[31: 0] RW		Configure the timer value of timer 4	32'b0

19.4.8 Timer 5 Timing Value Configuration Register

Table 169 Timer 5 Timing Value Configuration Register

bit access		Instructions	reset value
[31: 0]	RW	Configure the timer value of timer 5	32'b0

19.4.9 Timer 6 Timing Value Configuration Register

Table 170 Timer 6 Timing Value Configuration Register

bit access		Instructions	reset value
[31: 0]	RW	Configure the timer value of timer 6	32'b0

19.4.10 Timer 1 current count value register

Timer 1 current count value register

bit access		Instructions	reset value
[31:0]	RO	Read the current count value of timer 1;	32'b0

19.4.11 Timer 2 current count value register

Timer 2 current count value register

bit access		Instructions	reset value
[31:0]	RO	Read the current count value of timer 2;	32'b0

19.4.12 Timer 3 current count value register

Timer 3 current count value register

bit access	Instructions	reset value
[31:0] R0	Read the current count value of timer 3;	32'b0

19.4.13 Timer 4 current count value register

Timer 4 current count value register

bit access	Instructions	reset value
[31:0] R0	Read the current count value of timer 4;	32'b0

19.4.14 Timer 5 current count value register

Timer 5 current count value register

bit access	Instructions	reset value
[31:0] R0	Read the current count value of timer 5;	32'b0

19.4.15 Timer 6 current count value register

Timer 6 current count value register

bit access	Instructions	reset value
[31:0] R0	Read the current count value of timer 6;	32'b0

Winner Micro

20 Power Management Module

20.1 Function overview

The PMU realizes the switching of the working state of the chip hardware, as well as the power management during the state switching process, and provides timers, real-time clocks and

32K clocks.

20.2 Main Features

- ÿ Provide chip power control
- ÿ Provide timer function
- ÿ Provide real-time clock control
- ÿ Provide 32K RC oscillator calibration function
- ÿ Provide wake-up function;

20.3 Functional Description

20.3.1 Full Chip Power Control

PMU module controls the power switch of the chip, including 40M start-up circuit, BandGap, digital PLL, voltage detection circuit, digital circuit LDOs.

When the chip is powered on, the PMU module guides each module to turn on the power sequentially according to the preset power-on sequence;

When the software configuration register enters the sleep mode, each functional module is guided to turn off the power in turn according to the safe power-off sequence;

When the software configuration register enters the sleep mode, the clock, crystal start-up circuit and related power supply are turned off according to the sequence;

Provides three wake-up modes in sleep/sleep mode: Timer timed wakeup, RTC timed wakeup or by connecting the special WAKEUP pin

Pull high to wake up.

20.3.2 Low Power Mode

Two low-power modes can be selected by configuring the PMU register chip;

Standby mode:

In this mode, the power supply of the digital power domain will be turned off, and only the PMU module of the whole chip will work, providing wake-up and reset functions;

It consumes about 15uA. After the power is turned off, all data and content stored in the memory will be lost, and the firmware will be reloaded after waking up, which is equivalent to reloading

start up;

Sleep mode:

In this mode, the power of the digital power domain will be reserved, but the DPLL and the crystal start-up circuit will be turned off, and the clock will be cut off. At this time, the power consumption of the whole chip is about

About 1mA; the data and code stored in the memory will still be retained; the program will continue to run after waking up;

20.3.3 Wakeup Mode

PMU supports 3 wake-up modes, Timer wake-up, RTC wake-up and external IO wake-up.

Timer wake up

Before the software sets the sleep/sleep mode, configure the Timer0 module in the PMU and set the sleep time. When the system enters sleep mode,

When Timer0 reaches the sleep time, it will wake up the system and generate the corresponding Timer interrupt. After the system resumes operation, it is necessary to

Write '1' to the corresponding status bit in the status register to clear the interrupt status, otherwise, it will be woken up by the interrupt immediately after entering the sleep mode next time;

RTC wake up

Before the software sets the sleep/sleep mode, configure the RTC module in the PMU and set the sleep time. When the system enters sleep mode, when

When the RTC clock reaches the sleep time, it will wake up the system and give the corresponding RTC interrupt. After the system resumes operation, please update the interrupt register 0x14

Write '1' to the corresponding status bit in the middle to clear the interrupt status, otherwise, it will be woken up by the interrupt immediately after entering the sleep mode next time;

External IO wakeup

After software hibernation/sleep, the PMU will detect a specific Wakeup pin, the external controller can wake up the system by pulling this IO high, and give

Corresponding IO wake-up interrupt. The PMU no longer detects this IO state after leaving sleep mode. After the system resumes operation, please update the interrupt register 0x14

Write '1' to the corresponding status bit in the middle to clear the interrupt status, otherwise, it will be woken up by the interrupt immediately after entering the sleep mode next time;

20.3.4 Timer0 Timer

The timer enable signal and timing time are configured through the AHB register. First set the timing value, then set the timer enable BIT to start the timing

When the timer is reached, an interrupt is generated, and the software clears the interrupt flag by writing Bit0 of the status register.

20.3.5 Real time clock function

Reference Real Time Clock Module

20.3.6 32K clock source switching and calibration

W800 chip integrates 32K RC oscillator as the clock source of PMU module.

Due to the working environment and temperature changes, the output frequency of the 32K RC oscillator may change, resulting in timing deviation. Therefore, in the PMU mode

A 32K RC oscillator calibration function is introduced into the block, as well as a 32K clock switching function to correct for timing skew.

1) 32K clock source switching

The 32K clock can be switched from the 32K RC oscillator to the 40M clock divided by setting bit3 of the PS_CR register to 1.

to a 32K clock. However, when the chip enters sleep mode, bit3 will be automatically cleared to 0 because the 40M clock will be turned off. after waking up

If the firmware still needs to use the precise timing function, you need to reset bit3 to 1.

2) Calibration of 32K RC oscillator circuit

First set the bit2 of the PS_CR register to 0, and then set the bit2 of the PS_CR register to 1.

After calibration, the 32K RC oscillator will be relatively accurate. But if you want more accurate timing, it is recommended to use 40M clock frequency division

The obtained 32K clock will get the accurate RTC clock at this time, and the deviation will only be related to the crystal frequency deviation.

20.4 Register Description

20.4.1 Register List

Table 171 PMU register list

offset address	name	Abbreviated access	describe	reset value
0X0000	PMU Control Register PS_CR		RW is used to configure 32K calibration, configure 32K clock source, set the STANDBY function of the chip	0X0000_0002
0X0004	PMU Timer 0	TIMERO	RW Configure the timing value (in seconds), enable timing device	0X0000_0000
0X0008	Reserved			
0X0014	PMU interrupt source register INT_SRC		RW provides PMU interrupt flag	0X0000_0000

20.4.2 PMU Control Register

Table 172 PMU Control Register

bit access		Instructions	reset value

[31:11] RO			24'b0
[10]	RW	<p>Wake-up key interrupt polarity selection:</p> <p>0: The interrupt is set when the wake-up button is at a high level;</p> <p>1: The interrupt is set when the wake-up button is at a low level;</p> <p>Only valid in Sleep interrupt;</p>	1'b0
[9:6]	RW	<p>The minimum hold time for key wake-up:</p> <p>Unit: 128ms;</p>	4'd01
[5]	RW	<p>DLDO_CORE reference voltage source selection</p> <p>1: ABG</p> <p>0: DBG</p>	1'b1
[4]	RW	<p>32K oscillator circuit BYPASS signal</p> <p>Active high, set to 1, 32K is divided into 40M clock frequency. *2</p>	1'b0
[3]	RW	<p>RC 32K oscillator calibration circuit start switch;</p> <p>1'b0: calibration circuit reset;</p> <p>1'b1: start the calibration circuit;</p> <p>To start the calibration function, this bit needs to be set to 0 first and then set to 1.</p>	1'b0
[2]	RW	<p>Enable button trigger to enter sleep function</p> <p>0: Disable;</p> <p>1: Press the io_wakeup button to reach the threshold time in active mode , will trigger io_sleep_flag is interrupted and reported to the MCU.</p>	1'b0
[1]	RW	<p>Sleep enable signal, active high.</p> <p>1'b0: Chip wake-up state</p>	1'b1

		<p>1'b1: The chip enters the Sleep state</p> <p>If the WAKEUP pin is inactive and the TIMER0/1 interrupt wake-up is not configured, this register</p> <p>When valid, the chip enters the Sleep state;</p> <p>If the wake-up interrupt is generated, the chip will switch from Sleep state to wake-up state, and the wake-up condition is satisfied,</p> <p>This bit is automatically cleared to 0.</p> <p>Wake-up source: WAKEUP pin, TIMER0/TIMER1, RTC</p> <p>1) WAKEUP pin, active high; in order for the chip to enter the Sleep state, the WAKEUP pin must be in low level. To wake up, pull up the WAKEUP pin to generate a wake-up interrupt and make the chip leave Sleep state.</p> <p>2) TIMER0, timer wake-up interrupt.</p> <p>When the WAKEUP pin is low, TIMER0 sets the timing time and enables it. When the timing time is up, a wake-up will be generated.</p> <p>The wake-up interrupt causes the chip to leave the Sleep state.</p> <p>3) RTC, timed to wake up</p> <p>When the WAKEUP pin is low and the RTC time is up, a wake-up interrupt will be generated to make the chip leave the Sleep state.</p> <p>state</p>	
[0]	RW	<p>STANDBY enable signal, active high.</p> <p>1'b0: Chip wake-up state</p> <p>1'b1: The chip enters the STANDBY state</p> <p>If the WAKEUP pin is inactive and the TIMER0/1 interrupt wake-up is not configured, this register</p> <p>When valid, the chip enters the STANDBY state;</p> <p>If the wake-up interrupt is generated, the chip will switch from STANDBY state to wake-up state, and the wake-up condition is full enough, this bit is automatically cleared to 0.</p>	1'b0

	<p>Wake-up source: WAKEUP pin, TIMER0/TIMER1, RTC</p> <p>4) WAKEUP pin, active high; in order for the chip to enter the STANDBY state, the WAKEUP pin must be in low level. To wake up, pull up the WAKEUP pin to generate a wake-up interrupt and make the chip leave STANDBY state.</p> <p>5) TIMER0, timer wake-up interrupt.</p> <p>When the WAKEUP pin is low, TIMER0 sets the timing time and enables it. When the timing time is up, a wakeup will be generated. The wake-up interrupt causes the chip to leave the STANDBY state.</p> <p>6) RTC, timed to wake up</p> <p>When the WAKEUP pin is low and the RTC time is up, a wake-up interrupt will be generated, causing the chip to leave STANDBY status</p>	
--	--	--

20.4.3 PMU Timer 0

Table 173 PMU Timer 0 Register

bit access		Instructions	reset value
[31:17] RO	reserved		15'b0
[16]	RW	Timer0 enable bit 1'b0: Bit enable. 1'b1: enable;	1'b0
[15: 0]	RW	Timer0 timing value, unit: second	16'b0

20.4.4 PMU Interrupt Source Register

Table 174 PMU Interrupt Source Register

bit access		Instructions	reset value
[31:9]	R reserved		
[8]	RW	shows the current power-on status: 1'b0: Power-on or reset start 1'b1: wake up from sleep state, write 1 to clear	1'b0
[7]	RO reserved		1'b0
[6]	RO reserved		1'b0
[5]	RW RTC	timer interrupt flag bit: 1'b0: A timed interrupt is generated 1'b1: No timing interrupt is generated, write 1 to clear	1'b0
[4]	RW reserved		1'b0
[3]	RW reserved		1'b0
[2]	RW WAKEUP	pin wakeup interrupt flag 1'b0: No WAKEUP wake-up interrupt is generated 1'b1: WAKEUP wake-up interrupt is generated, write 1 to clear	1'b0
[1]	RW reserved		1'b0
[0]	RW Timer0	timer interrupt flag bit: 1'b0: No Timer0 interrupt is generated 1'b1: Timer0 interrupt is generated, write 1 to clear	1'b0

Winner Micro

21 Real time clock module

21.1 Function overview

The RTC is a BCD counter/timer provided by the PMU module. The two 32-bit registers contain seconds, minutes, hours, days, months, and years.

The decimal format representation of the hexadecimal code (BCD) can automatically correct the months of 28, 29 (leap year), 30, and 31 days.

Under the corresponding software configuration, the RTC can not only provide the clock calendar function, but also can be used as a timer, when the timer reaches the set time

An RTC interrupt is then generated, which can be used to wake up the system from a sleep state.

The RTC module has two clock sources that can be configured: 40M clock frequency division and internal 32K clock. Which can be used by software configuration during normal work

clock source; only 32K clocks can be used in sleep state. If the RTC clock source is divided by 40M clock in normal working state, then

After entering the sleep state, it will automatically switch to the 32K clock, and the system will keep using the 32K clock after being woken up. So as long as the power supply voltage

Within the working range, the RTC module will not stop working regardless of whether the module is in a normal working state or a sleep state.

21.2 Main Features

- ÿ Provide timing function
- ÿ Provide timing function
- ÿ Provide timed interrupt
- ÿ Interrupt to wake up the system

21.3 Functional Description

21.3.1 Timing function

The initial value of day, hour, minute and second can be configured in RTC configuration register 1, and the initial value of year and month can be configured in RTC configuration register 2.

The timekeeping function is enabled in RTC Configuration Register 2.

After the RTC timing function is enabled, read the RTC configuration register 1 to get the current day, hour, minute and second values.

Register 2 can get the current year and month value.

21.3.2 Timing function

The day, hour, minute, and second timing values can be configured in RTC configuration register 1, and the year and month timing values can be configured in RTC configuration register 2.

The RTC configuration register 1 enables the timing function.

When the RTC timer reaches the timing time, an RTC interrupt will be generated. At this time, set the RTC interrupt bit of the PMU interrupt source register to 1 to clear except the interrupted state.

When the system enters sleep mode, the interrupt generated by the RTC timer will wake up the system.

21.4 Register Description

21.4.1 Register List

The RTC module has a total of two 32-bit dedicated registers, and the RTC interrupt status needs to query the PMU interrupt source register.

Table 175 RTC register list

offset address	name	Abbreviated access	describe	reset value
0X000C	RTC Configuration Register 1	RTC_R1 RW	Configure RTC day, hour, minute and second value, configure enable timing 0X0000_0000	
0X0010	RTC Configuration Register 2	RTC_R2 RW	Configure RTC year and month value, configure enable timing 0X0000_0000	

21.4.2 RTC Configuration Register 1

Table 176 RTC Configuration Register 1

bit access		Instructions	reset value
[31]	RW	RTC timer interrupt function enable	1'b0

		1'b0: Disable 1'b1: enable	
[30:29]		reserve	
[28:24] RW		Day start value/day time value	5'b0
[23:21]		reserve	
[20:16] RW		Hour initial value/hour timing value	5'b0
[15:14]		reserve	
[13:8]		reserve	
[7:6]		reserve	
[5 : 0] RW		Second initial value/second timing value	6'b0

21.4.3 RTC Configuration Register 2

Table 177 RTC Configuration Register 2

bit access		Instructions	reset value
[31:17]		reserve	
[16]	RW	RTC timing function enable bit 1'b0: Disable 1'b1: enable	1'b0
[15]		reserve	
[14:8] RW		Beginning of the year value/yearly fixed value	7'b0
[7:4]		reserve	
[3 : 0] RW		Monthly value/monthly fixed value	4'b0

22 Watchdog module

22.1 Function overview

Implement the "watchdog" function. Designed for global reset in case of system crash.

"Watchdog" will generate a periodic interrupt. After the interrupt is generated, the system software will clear its interrupt flag. If it exceeds the set time, it will not be cleared.

Otherwise, a hard reset signal will be generated to reset the system.

22.2 Main Features

- ÿ Provide timing function
- ÿ Provide reset function
- ÿ Provide timed interrupt

22.3 Functional Description

22.3.1 Timing function

After setting the timing value to the register WD_LD, set the BIT0 of WDG_CTRL to 1 to start the timer, and the WDG module will generate the timer when the timing is up.

When a timed interrupt occurs, the notification program is processed. If the BIT0 of the register WD_CLR is not cleared, a timed interrupt will be generated periodically.

The value of WD_LD is based on the APB clock unit, and the APB clock is divided from the 160M clock.

22.3.2 Reset function

After setting the chip timing value WD_LD, start the timing and reset function (set BIT1/BIT0 of WDG_CTRL), and the WDG module starts the reverse operation.

Timing, when the timing time is up, WDG will generate a timing interrupt. At the same time, if the BIT0 of WD_CLR is not cleared, the chip will

The reset signal is generated in the next cycle.

22.4 Register Description

22.4.1 Register List

Table 178 WDG Register List

offset address	name	Abbreviated access	describe	reset value
0X0000	WDG timing load register WD_LD	RW	configures the timing value for repeated loading	0XFFFF_FFFF
0X0004	WDG current value register WD_VAL	RO	gets the value of the current timer	0XFFFF_FFFF
0X0008	WDG Control Register WD_CTRL	RW	Control Register	0X0000_0000
0X000C	WDG Interrupt Clear Register WD_CLR	WO	Interrupt Clear Register	0X0000_0000
0X0010	WDG interrupt source register WD_SRC	RO	Interrupt Source Register	0X0000_0000
0X0014	WDG Interrupt Output Register WD_STATE	RO	Interrupt Output Status Register	0X0000_0000

22.4.2 WDG Timing Value Load Register

Table 179 WDG Timing Value Load Register

bit access		Instructions	reset value
[31: 0] RW		<p>Configure timing values for repeated loading</p> <p>The value of this register is counted in APB clocks.</p> <p>For example: if the APB clock is 40MHZ, the maximum duration of the timing value is about 107s, that is 0xFFFFFFFF/40000000</p>	32'hffff_ffff

22.4.3 WDG Current Value Register

Table 180 WDG current value register

bit access		Instructions	reset value
[31:0] RO		<p>Get the value of the current timer</p> <p>To calculate the remaining time, just read the current value.</p> <p>To calculate the elapsed time, simply subtract the value of register WD_VAL from the value of register WD_LD</p>	32'hffff_ffff

22.4.4 WDG Control Register

Table 181 WDG Control Register

bit access		Instructions	reset value
[31:2]		reserve	30'h0
[1]	RW	<p>reset enable bit</p> <p>1'b0: When the WDG reset condition occurs, no reset signal is generated</p> <p>1'b1: When the WDG reset condition occurs, the reset signal is generated</p>	1'b0
[0]	RW	<p>timing enable bit</p> <p>1'b0: Timer not working</p> <p>1'b1: The timer works and generates periodic interrupts</p>	1'b0

22.4.5 WDG Interrupt Clear Register

Table 182 WDG Interrupt Clear Register

bit access		Instructions	reset value
[31:1]		reserve	31'h0

[0]	WO	Interrupt status clear bit, write any value to clear the current interrupt status	1'b0
-----	----	---	------

22.4.6 WDG Interrupt Source Register

Table 183 WDG Interrupt Source Register

bit access		Instructions	reset value
[31:1]		reserve	31'h0
[0]	RO	The interrupt source register, the timer function is turned on, will generate the interrupt at the same time	1'b0

22.4.7 WDG Interrupt Status Register

Table 184 WDG Interrupt Status Register

bit access		Instructions	reset value
[31:1]		reserve	31'h0
[0]	RO	Interrupt output status register. This interrupt is not generated after the timer is turned off, but WD_SRC may be 1	1'b0

23 PWM controller

23.1 Function overview

PWM is a method of digitally encoding analog signal levels. Through the use of a high resolution counter, the duty cycle of the square wave is modulated with

to encode the level of a specific analog signal. The PWM signal is still digital because at any given moment, the full-scale

The current supply is either completely present (ON) or completely absent (OFF). A voltage or current source is a repetitive pulse of ON or OFF.

The pulse sequence is added to the simulated load. When it is on, the DC power supply is applied to the load, and when it is off, the power supply is disconnected.

when. Any analog value can be encoded using PWM as long as the bandwidth is sufficient.

23.2 Main Features

- ÿ Support 2-channel input signal capture function (two channels of PWM0 and PWM4)
- ÿ Input signal capture function supports interrupt interactive mode and DMA transfer mode; DMA mode supports word-by-word operation
- ÿ Support 5-channel PWM signal generation function
- ÿ 5-channel PWM signal generation supports one-shot generation mode and auto-load mode
- ÿ Support 5-channel braking function
- ÿ PWM output frequency range: 3Hz~160kHz
- ÿ The maximum precision of the duty cycle: 1/256, the width of the counter inserted in the dead zone: 8bit
- ÿ Support channel 0 channel 1 synchronization function, support channel 2 channel 3 synchronization function
- ÿ Support complementary and non-complementary modes of channel 0, channel 1, and complementary and non-complementary modes of channel 2 and channel 3
- ÿ Support 5-channel sync function

23.3 Functional Description

23.3.1 Input Signal Capture

The PWM controller supports the signal capture function of two channels, and the capture of channel 0 can be activated by setting Bit24 of the PWM_CTL register

function, the capture function of channel 4 can be activated by setting Bit1 of the PWM_CAP2CTL register. The level of the captured signal can also be

to set whether to flip the function. After the channel captures the corresponding signal, the capture number is updated to the corresponding capture register PWM_CAPDAT (pass

channel 0 capture number) and PWM_CAP2DAT (channel 4 capture number).

23.3.2 DMA Transfer Captures

After channel 0 or channel 4 enables the capture function, the count of the capture register can be quickly transferred to the memory through the DMA channel to speed up the user process.

23.3.3 Support for single-shot and autoload modes

The five output channels of the PWM controller all support one-shot output mode and auto-load mode. In single-load mode, the channel outputs the specified cycle

After the waveform, the PWM wave will no longer be output; in the automatic loading mode, after the channel outputs the specified cycle waveform, the cycle will be automatically reloaded number, so as to continue to generate PWM waves.

23.3.4 Multiple Output Modes

The PWM controller supports independent output mode, that is, each channel outputs independently without interfering with each other; it supports dual-channel synchronous mode, that is, one channel

The output is exactly the same as the output of another channel; the five-channel sync mode is supported, and the output of channel 1 to channel 4 is completely the same as the output of channel 0.

It supports dual-channel complementary output, that is, the waveform output by one channel is completely opposite to the waveform output by the other channel; supports complementary mode

Commonly used dead zone settings, the dead zone length can be set up to 256 clock cycles; support braking mode, when the braking port detects the specified power

After leveling, the output channel will output the set braking level.

A variety of output modes are flexible and configurable, which can satisfy users' various application scenarios related to PWM.

23.4 Register Description

23.4.1 PWM Register List

Table 185 PWM register list

offset address name		abbreviation	access	describe	reset value
0X0000 Clock divider register_01	PWM_CLKDIV01	V01	RW	Divide the clock of channel 0 and channel 1 0X0000_0000	
0X0004 Clock frequency division register_23	PWM_CLKDIV23	V23	RW	Divide the clock of channel 2 and channel 3 0X0000_0000	
0X0008 Control register	PWM_CTL		RW	is used to configure or control some configurable items 0X0000_0000	
0X000C Period register	PWM_PERIOD		RW	is used to set the period from channel 0 to channel 4 0X0000_0000	
0X0010 Cycle number register	PWM_PNUM		RW	is used to set the signal generation of channel 0 to channel 4 into cycles	0X0000_0000
0X0014 Compare register	PWM_CMPDAT		RW	is used to store the comparison value of channel 0 to channel 4 to produce different duty cycles	0X0000_0000
0X0018 Dead zone control register	PWM_DTCTL		RW	is used to configure or control the configurable set item	0X0000_0000
0X001C Interrupt Control Register	PWM_INTEN		RW	is used to enable and control related interrupts 0X0000_0000	
0X0020 Interrupt Status Register	PWM_INTSTS		RW	is used to query the status of related interrupts	0X0000_0000
0X0024 Channel 0 capture register	PWM_CAPDAT		RO	is used to capture and count the rising edge and falling edge to channel 0	0X0000_0000
0X0028 Brake control register	PWM_BRKCTL		RW	is used to control the braking mode	0X0000_0000
0X002C Clock frequency division register_4	PWM_CH4_reg1	V4	RW	Divide the clock of channel 4	0X0000_0000

0X0030	Channel 4 control register_1 PWM_CH4_reg2 RW Set related configuration items of channel 4 0X0000_0000			
0X0034	Channel 4 capture register PWM_CAP2DAT	RO is used to capture and count the rising to channel 4 edge and falling edge		0X0000_0000
0X0038	Channel 4 control register_2 PWM_CAP2CTL RW Set related configuration items of channel 4 0X0000_0000			

23.4.2 Clock divider register_01

Table 186 PWM clock divider register_01

bit access		Instructions	reset value
[31:16] RW		CLKDIV1 CH1 frequency division counter The frequency division is determined by the counter value Note: The frequency division range is (0~65535). If frequency division is not required, enter 0 or 1.	16'h0
[15:0] RW		CLKDIV0 CH0 frequency division counter Same as CH1	16'h0

23.4.3 Clock divider register_23

Table 187 PWM clock divider register_23

bit access		Instructions	reset value
[31:16] RW		CLKDIV3 CH3 frequency division counter Same as CH1	16'h0
[15:0] RW		CLKDIV2	16'h0

		CH2 frequency division counter Same as CH1	
--	--	---	--

23.4.4 Control Register

Table 188 PWM Control Register

bit access		Instructions	reset value
[31:27] RW		CNTEN Counter count enable 1'b0: stop counting 1'b1: start counting Note: Each bit controls each channel separately, and controls CH4, CH3, CH2, CH1 and CH0 in sequence from high to low	5'b0
[26]	--	reserve	1'b0
[25]	RW	CAPINV Capture reverse enable flag 1'b0: Inverse of input signal in capture mode is invalid 1'b1: The input signal in the capture mode is valid in reverse, and the input signal is reversed	1'b0
[twenty four]	RW	CPEN Capture function enable flag 1'b0: CH0 capture function is invalid, RCAPDAT and FCAPDAT values will not be updated; 1'b1: CH0 capture function is valid, capture and latch the PWM counter, respectively store in RCAPDAT (rising edge latch) and FCAPDAT (falling edge latch)	1'b0
[23:22] RW		CNTTYPE3	2'b0

		<p>CH3 Counter counting method</p> <p>2'b00: edge-aligned mode (counter counting is incremented, only for capture mode)</p> <p>2'b01: edge-aligned mode (counter counts down, only for PWM mode)</p> <p>2'b10: Center-aligned mode (PWM mode only)</p> <p>Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement Way.</p>	
[21:20] RW		<p>CNTTYPE2</p> <p>CH2 Counter counting method</p> <p>Same as CH3</p>	2'b0
[19:18] RW		<p>CNTTYPE1</p> <p>CH1 Counter counting method</p> <p>Same as CH3</p>	2'b0
[17:16] RW		<p>CNTTYPE0</p> <p>CH0 Counter counting method</p> <p>Same as CH3</p>	2'b0
[15:14] RW		<p>TWOSYNCEN</p> <p>2-channel sync mode enable signal</p> <p>1'b0: 2-channel synchronization not allowed</p> <p>1'b1: Enable 2-channel synchronization,</p> <p>PWM_CH0 and PWM_CH1 have the same phase, and the phase is determined by PWM_CH0; PWM_CH2</p> <p>Has the same phase as PWM_CH3, and the phase is determined by PWM_CH2</p> <p>15bit control CH3 and CH2</p>	2'b0

		14bit control CH1 and CH0	
[13]	-- reserve		1'b0
[12]	RW	<p>POEN</p> <p>PWM pin output enable bit</p> <p>1'b0: PWM pin is set to output state</p> <p>1'b1: PWM pin is tri-stated</p> <p>Note: only for CH0</p>	1'b0
[11:8] RW		<p>CNTMODE</p> <p>PWM generation loop method</p> <p>1'b0: single shot mode</p> <p>1'b1: Autoload mode</p> <p>Note: During the change of CNTMODE, PWM_CMPDAT returns to zero; each bit controls each channel separately, from high</p> <p>To low control PW3, PW2, PW1 and PW0 in turn</p>	4'h0
[7]	-- reserve		1'b0
[6]	RW	<p>ALLSYNCEN</p> <p>All-channel sync mode enable signal</p> <p>1'b0: All channels are not allowed to synchronize</p> <p>1'b1: All channels are allowed to synchronize, PWM_CH0, PWM_CH1, PWM_CH2 and PWM_CH3 have</p> <p>The same phase, and the phase is determined by PWM_CH0</p>	1'b0
[5:2] RW		<p>PINV</p> <p>PWM output signal polarity enable</p> <p>1'b0: PWM output polarity reversal disabled</p>	4'h0

		<p>1'b1: PWM output polarity inversion enable</p> <p>Note: Each bit controls each channel separately, and controls PW3, PW2, PW1 and PW0 in turn from high to low</p>	
[1:0] RW		<p>OUTMODE</p> <p>output mode</p> <p>1'b0: Non-complementary mode for every two channels</p> <p>1'b1: Complementary mode for every two channels</p> <p>BIT1 controls CH2 and CH3</p> <p>BIT0 controls CH0 and CH1</p>	2'b0

23.4.5 Period Register

Table 189 PWM Period Register

bit access		Instructions	reset value
[31:24] RW		<p>PERIOD3</p> <p>CH3 period register value (Note: period cannot be greater than 255)</p> <p>"Edge-aligned mode (counter counts down)":</p> <ul style="list-style-type: none"> ÿ PERIOD register value, period value is (PERIOD + 1) ÿ Duty cycle = (CMP+1)/(PERIOD + 1) ÿ CMP>=PERIOD: PWM output is fixed high ÿ CMP<PERIOD: PWM low level width is (PERIOD-CMP), high level width is (CMP+1) ÿ CMP=0: PWM low level width is PERIOD, high level width is 1; <p>"Center Alignment Mode":</p> <ul style="list-style-type: none"> ÿ PERIOD register value: period is 2*(PERIOD+1) 	8'h0

		<p>ÿ Duty cycle=(2*CMP+1)/(2*(PERIOD+1))</p> <p>ÿ CMP>PERIOD: PWM is continuously high</p> <p>ÿ CMP<=PERIOD: PWM low level=2*(PERIOD-CMP)+1, High level=(2*CMP)+1</p> <p>ÿ CMP=0: PWM low level width is 2*PERIOD+1, high level width is 1.</p> <p>Note: In "center-aligned mode", the number of cycles should not be 255.</p> <p>No matter which alignment mode is selected, the channel period is determined by the division number (N) and the number of periods (P),</p> <p>That is: the input clock is 40MHz, the clock frequency f_div after frequency division is: $f_{div} = 40\text{MHz}/N$, N is the division Frequency (16bit). The output frequency f_output is: $f_{output} = f_{div} / P$, where P is the number of cycles.</p> <p>Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement Way.</p>	
[23:16] RW		<p>PERIOD2</p> <p>CH2 period register value (note: period cannot be greater than 255)</p> <p>Same as PERIOD3</p>	8'h0
[15:8] RW		<p>PERIOD1</p> <p>CH1 period register value (Note: period cannot be greater than 255)</p> <p>Same as PERIOD3</p>	8'h0
[7:0] RW		<p>PERIOD0</p> <p>CH0 period register value (Note: period cannot be greater than 255)</p> <p>Same as PERIOD3</p>	8'h0

23.4.6 Cycle Number Register

Table 190 PWM Period Number Register

bit access		Instructions	reset value
[31:24] RW		<p>PNUM3</p> <p>PWM3 Generation Cycles</p> <p>Set the number of PWM3 cycles PNUM3, when the PWM generates PNUM3 PWM signals, stop generating signals.</p> <p>number, trigger the interrupt and set the interrupt status word at the same time</p>	8'h0
[23:16] RW		<p>PNUM2</p> <p>Number of PWM2 generation cycles</p> <p>Same as PNUM3</p>	8'h0
[15:8] RW		<p>PNUM1</p> <p>Number of PWM1 generation cycles</p> <p>Same as PNUM3</p>	8'h0
[7:0] RW		<p>PNUM0</p> <p>PWM0 generation cycle number</p> <p>Same as PNUM3</p>	8'h0

23.4.7 Compare Register

Table 191 PWM Compare Register

bit access		Instructions	reset value
[31:24] RW		<p>CMP3</p> <p>PWM3 compare register value</p>	8'h0

		<p>"Edge-aligned mode (counter counts down)":</p> <ul style="list-style-type: none"> ÿ PERIOD register value, period value is (PERIOD + 1) ÿ Duty cycle = (CMP+1)/(PERIOD + 1) ÿ CMP>=PERIOD: PWM output fixed bit high ÿ CMP<PERIOD: PWM low level width is (PERIOD-CMP), high level width is (CMP+1) ÿ CMP=0: PWM low level width is PERIOD, high level width is 1; <p>"Center Alignment Mode":</p> <ul style="list-style-type: none"> ÿ PERIOD register value: period is 2*(PERIOD+1) ÿ Duty ratio=(2*CMP+1)/2*(PERIOD+1) ÿ CMP>PERIOD: PWM is continuously high ÿ CMP<=PERIOD: PWM low level=2*(PERIOD-CMP)+1, high level=(2*CMP)+1 ÿ CMP=0: PWM low level width is 2*PERIOD+1, high level width is 1. <p>No matter which alignment mode is selected, the channel period is determined by the division number (N) and the number of periods (P), namely:</p> <p>The input clock is 40MHz, the clock frequency f_div after frequency division is: $f_{div} = 40\text{MHz}/N$, N is the frequency division number (16bit). The output frequency f_output is: $f_{output} = f_{div} / P$, where P is the number of cycles.</p> <p>Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement Way.</p>	
[23:16] RW		CMP2 PWM2 compare register value Same as CMP3	8'h0
[15:8] RW		CMP1 PWM1 compare register value	8'h0

		Same as CMP3	
[7:0] RW		CMP0 PWM0 compare register value Same as CMP3	8'h0

23.4.8 Dead Time Control Register

Table 192 PWM Dead Time Control Register

bit access		Instructions	reset value
[31:22]	-- reserve		10'h0
[twenty one]	RW	<p>DTEN23</p> <p>Whether channel 2 and channel 3 can insert deadband valid flag</p> <p>Insert deadband valid signal is valid only after the complementary mode of the channel is turned on. And, if a valid signal is inserted</p> <p>If it is 0, the complementary signal output by the two channels has no dead zone insertion.</p> <p>1'b0: Invalid insertion dead zone</p> <p>1'b1: Insert dead zone valid</p>	1'b0
[20]	RW	<p>DTEN01</p> <p>Can channel 0 and channel 1 insert deadband valid flags?</p> <p>Same as DTEN23</p>	1'b0
[19:18]	-- reserve		2'b0
[17:16] RW		<p>DTDIV</p> <p>Dead time clock divider control</p> <p>2'b00: Dead time clock equal to base clock (40MHz)</p>	2'b0

		2'b01: Dead-band clock equal to base clock (40MHz) divided by two 2'b10: Dead-band clock equal to base clock (40MHz) divided by four 2'b11: Dead-band clock equal to base clock (40MHz) divided by eight	
[15:8] RW		DTCNT23 Dead time interval for channel 3 and channel 2 8bit determines the dead zone interval value, and the dead zone clock is determined by DTDIV	8'h0
[7:0] RW		DTCNT01 Dead time interval for channel 1 and channel 0 8bit determines the dead zone interval value, and the dead zone clock is determined by DTDIV	8'h0

23.4.9 Interrupt Control Register

Table 193 PWM Interrupt Control Register

bit access		Instructions	reset value
[31:8]	-- reserved		24'h0
[7]	RW	DMA_request_EN DMA_request enable 1'b0: DMA_request is invalid 1'b1: DMA_request is valid	1'b0
[6]	RW	FLIEN Falling edge buffer interrupt enable bit 1'b0: Falling edge buffer interrupt invalid 1'b1: Falling edge buffer interrupt is valid	1'b0

		Note: For CH0	
[5]	RW	<p>RLIEN</p> <p>Rising edge buffer interrupt enable bit</p> <p>1'b0: Rising cache interrupt invalid</p> <p>1'b1: Rising edge buffer interrupt is valid</p> <p>Note: For CH0</p>	1'b0
[4:0] RW		<p>PIEN</p> <p>PWM Period Interrupt Enable Bit</p> <p>1'b0: Period interrupt is invalid</p> <p>1'b1: Periodic interrupt is valid</p> <p>Note: When the counter counts to 0 and the number of PWM cycles meets PWM_PNUM, an interrupt is triggered.</p>	5'b0

23.4.10 Interrupt Status Register

Table 194 PWM Interrupt Status Register

bit access		Instructions	reset value
[31:10]	-- reserve		12'h0
[9]	RW	<p>OVERFL</p> <p>Counter overflow flag</p> <p>1'b0: Capture mode, the counter does not overflow during the counting process</p> <p>1'b1: Capture mode, counter overflows during counter counting</p> <p>Note: When the user clears CFLIF or CRLIF, this bit is also cleared at the same time</p>	1'b0
[8]	RW	FLIFOV	1'b0

		Falling edge delay interrupt flags overrun status 1'b0: When CFILF is 1, no falling edge delay interrupt is generated 1'b1: When CFILF is 1, another falling edge delay interrupt occurs Note: When the user clears CFILF, this bit is also cleared at the same time	
[7]	RW	RLIFOV Rising edge delay interrupt flag overrun status 1'b0: When CRILF is 1, no rising edge delay interrupt is generated 1'b1: When CRILF is 1, another rising edge delay interrupt occurs Note: When the user clears CRILF, this bit is also cleared at the same time	1'b0
[6]	RW	CFLIF Capture falling edge interrupt flag 1'b0: No falling edge captured 1'b1: When a falling edge is captured, this bit is set to 1 Note: By writing 1, the flag is cleared; Note: For CH0	1'b0
[5]	RW	CRLIF Capture rising edge interrupt flag 1'b0: no rising edge captured 1'b1: When a rising edge is captured, this bit is set to 1 Note: By writing 1, the flag is cleared; Note: For CH0	1'b0
[4:0]	RW	PIF	5'b0

		<p>PWM Period Interrupt Flag</p> <p>When PWM generates a PWM signal with a specified period, the flag is set to 1; write 1 through software to clear the flag</p> <p>Note: Each bit identifies each channel, and controls PW4, PW3, PW2, PW1 and PW0 in sequence from high to low</p>	
--	--	---	--

23.4.11 Channel 0 Capture Register

Table 195 PWM Channel 0 Capture Register

bit access		Instructions	reset value
[31:16] RO		<p>PWM_FCAPDAT</p> <p>Capture falling edge register</p> <p>When there is a falling edge of the input signal, the current counter value is stored</p>	16'h0
[15:0] RO		<p>PWM_RCAPDAT</p> <p>Capture rising edge register</p> <p>When there is a rising edge of the input signal, the current counter value is stored</p>	16'h0

23.4.12 Brake Control Register

Table 196 PWM Brake Control Register

bit access		Instructions	reset value
[31:16]	-- reserve		16'h0
[15:11] RW		<p>BRKCTL</p> <p>Brake Mode Enable</p> <p>1'b0: Braking mode disabled</p> <p>1'b1: Brake mode activated</p>	5'b0

		[7:3] Corresponding to CH4, CH3, CH2, CH1 and CH0 respectively	
[10:8]	-- reserve		3'b0
[7:3] RW		<p>BKOD</p> <p>Brake output control register</p> <p>1'b0: When the braking mode is valid, the PWM output is low level</p> <p>1'b1: PWM output high level when braking mode is valid</p> <p>[7:3] Corresponding to CH4, CH3, CH2, CH1 and CH0 respectively</p>	5'b0
[2:0]	-- reserve		3'b0

23.4.13 Clock divider register_4

Table 197 PWM clock divider register_4

bit access		Instructions	reset value
[31:16] RW		<p>CLKDIV4</p> <p>CH4 Frequency division counter</p> <p>The frequency division is determined by the counter value</p> <p>Note: The frequency division range is (0~65535). If frequency division is not required, enter 0 or 1.</p>	16'h0
[15:8] RW		<p>PERIOD4</p> <p>CH4 period register value (Note: period cannot be greater than 255)</p> <p>"Edge-aligned mode (counter counts down)":</p> <ul style="list-style-type: none"> ÿ PERIOD register value, period value is (PERIOD + 1) ÿ Duty cycle = (CMP+1)/(PERIOD + 1) ÿ CMP>=PERIOD: PWM output fixed bit high 	8'h0

		<p>ÿ CMP<PERIOD: PWM low level width is (PERIOD-CMP), high level width is (CMP+1)</p> <p>ÿ CMP=0: PWM low level width is PERIOD, high level width is 1;</p> <p>"Center Alignment Mode":</p> <p>ÿ PERIOD register value: period is 2*(PERIOD+1)</p> <p>ÿ Duty cycle=(2*CMP+1)/(2*(PERIOD+1))</p> <p>ÿ CMP>PERIOD: PWM is continuously high</p> <p>ÿ CMP<=PERIOD: PWM low level=2*(PERIOD-CMP)+1, high level=(2*CMP)+1</p> <p>ÿ CMP=0: PWM low level width is 2*PERIOD+1, high level width is 1.</p> <p>Note: In "center-aligned mode", the number of cycles should not be 255.</p> <p>No matter which alignment mode is selected, the channel period is determined by the division number (N) and the number of periods (P),</p> <p>That is: the input clock is 40MHz, the clock frequency f_div after frequency division is: $f_{div} = 40\text{MHz}/N$, N is the division Frequency (16bit). The output frequency f_output is: $f_{output} = f_{div} / P$, where P is the number of cycles.</p> <p>Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement Way.</p>	
[7:0] RW		<p>CH4 Generation Cycles</p> <p>Set the number of PWM4 cycles to PNUM4, after the PWM generates PNUM4 PWM signals,</p> <p>Stop generating a signal while triggering an interrupt and setting the interrupt status word</p>	8'h0

23.4.14 Channel 4 Control Register_1

Table 198 PWM Channel 4 Control Register_1

bit access		Instructions	reset value
[31:16]	-- reserve		16'h0

[15:8] RW	CMP4 CH4 period register value "Edge-aligned mode (counter counts down)": ÿ PERIOD register value, period value is (PERIOD + 1) ÿ Duty cycle = (CMP+1)/(PERIOD + 1) ÿ CMP>=PERIOD: PWM output fixed bit high ÿ CMP<PERIOD: PWM low level width is (PERIOD-CMP), high level width is (CMP+1) ÿ CMP=0: PWM low level width is PERIOD, high level width is 1; "Center Alignment Mode": ÿ PERIOD register value: period is 2*(PERIOD+1) ÿ Duty ratio=(2*CMP+1)/2*(PERIOD+1) ÿ CMP>PERIOD: PWM is continuously high ÿ CMP<=PERIOD: PWM low level=2*(PERIOD-CMP)+1, high level=(2*CMP)+1 ÿ CMP=0: PWM low level width is 2*PERIOD+1, high level width is 1. No matter which alignment mode is selected, the channel period is determined by the division number (N) and the number of periods (P), namely: The input clock is 40MHz, the clock frequency f_div after frequency division is: $f_{div} = 40\text{MHz}/N$, N is the frequency division number (16bit). The output frequency f_output is: $f_{output} = f_{div} / P$, where P is the number of cycles. Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement Way.	8'h0
[7:5]	-- reserve	3'b0
[4:3] RW	CNTTYPE4 CH4 Counter counting method	2'b0

		<p>2'b00: edge-aligned mode (counter counting is incremented, only for capture mode)</p> <p>2'b01: edge-aligned mode (counter counts down, only for PWM mode)</p> <p>2'b10: Center-aligned mode (PWM mode only)</p> <p>Note: In PWM mode, when the counter is set to edge-aligned mode, you need to set the counting method to decrement Way.</p>	
[2]	-- reserve		1'b0
[1]	RW	<p>CNTMODE4</p> <p>CH4 generation cycle method</p> <p>1'b0: single shot mode</p> <p>1'b1: Autoload mode</p> <p>Note: During CNTMODE changing, PWM_CMPDAT returns to zero</p>	1'b0
[0]	RW	<p>PINV4</p> <p>CH4 output signal polarity enable</p> <p>1'b0: PWM output polarity reversal disabled</p> <p>1'b1: PWM output polarity inversion enable</p>	1'b0

23.4.15 Channel 4 Capture Register

Table 199 PWM Channel 4 Capture Register

bit access		Instructions	reset value
[31:16] RO		<p>PWM_FCAP2DAT</p> <p>Capture falling edge register</p> <p>When there is a falling edge of the input signal, the current counter value is stored</p>	16'h0

[15:0] RO		PWM_RCAP2DAT Capture rising edge register When there is a rising edge of the input signal, the current counter value is stored	16'h0
-----------	--	--	-------

23.4.16 Channel 4 Control Register_2

Table 200 PWM Channel 4 Control Register_2

bit access		Instructions	reset value
[31:11]	-- reserve		21'h0
[10]	RW	DMA_request2_mask DMA_request2 enable 1'b0: DMA_request2 is invalid 1'b1: DMA_request2 is valid Note: only for CH4	1'b0
[9]	RW	FLIEN2 Falling edge buffer interrupt enable bit 1'b0: Falling edge buffer interrupt invalid 1'b1: Falling edge buffer interrupt is valid Note: only for CH4	1'b0
[8]	RW	RLIEN2 Rising edge buffer interrupt enable bit 1'b0: Rising cache interrupt invalid 1'b1: Rising edge buffer interrupt is valid Note: only for CH4	1'b0

[7]	RW	<p>OVERFL2</p> <p>Counter overflow flag</p> <p>1'b0: Capture mode, the counter does not overflow during the counting process</p> <p>1'b1: Capture mode, counter overflows during counter counting</p> <p>Note: When the user clears CFLIF or CRLIF, this bit is also cleared at the same time</p> <p>Note: only for CH4</p>	1'b0
[6]	RW	<p>FLIFOV2</p> <p>Falling edge delay interrupt flags overrun status</p> <p>1'b0: When CFILF is 1, no falling edge delay interrupt is generated</p> <p>1'b1: When CFILF is 1, another falling edge delay interrupt occurs</p> <p>Note: When the user clears CFILF, this bit is also cleared at the same time</p> <p>Note: only for CH4</p>	1'b0
[5]	RW	<p>RLIFOV2</p> <p>Rising edge delay interrupt flag overrun status</p> <p>1'b0: When CRILF is 1, no rising edge delay interrupt is generated</p> <p>1'b1: When CRILF is 1, another rising edge delay interrupt occurs</p> <p>Note: When the user clears CRILF, this bit is also cleared at the same time</p> <p>Note: only for CH4</p>	1'b0
[4]	RW	<p>CFLIF2</p> <p>Capture falling edge interrupt flag</p> <p>1'b0: No falling edge captured</p> <p>1'b1: When a falling edge is captured, this bit is set to 1</p>	1'b0

		<p>Note: By writing 1, this flag is cleared</p> <p>Note: only for CH4</p>	
[3]	RW	<p>CRLIF2</p> <p>Capture rising edge interrupt flag</p> <p>1'b0: no rising edge captured</p> <p>1'b1: When a rising edge is captured, this bit is set to 1</p> <p>Note: By writing 1, this flag is cleared</p> <p>Note: only for CH4</p>	1'b0
[2]	RW	<p>POEN2</p> <p>PWM pin output enable bit</p> <p>1'b0: PWM pin is set to output state</p> <p>1'b1: PWM pin is tri-stated</p> <p>Note: only for CH4</p>	1'b0
[1]	RW	<p>CPEN2</p> <p>Capture function enable flag</p> <p>1'b0: CH4 capture function is invalid, RCAPDAT and FCAPDAT values will not be updated;</p> <p>1'b1: CH4 capture function is valid, capture and latch the PWM counter, respectively store in RCAPDAT (rising edge latch) and FCAPDAT (falling edge latch)</p> <p>Note: only for CH4</p>	1'b0
[0]	RW	<p>CAPINV2</p> <p>Capture reverse enable flag</p> <p>1'b0: Inverse of input signal in capture mode is invalid</p>	1'b0

		<p>1'b1: The input signal in the capture mode is valid in reverse, and the input signal is reversed</p> <p>Note: only for CH4</p>	
--	--	---	--

24 QFLASH Controller

24.1 Function overview

W800 has a built-in QFLASH controller, which provides QFLASH read, write and erase operations in bus mode, and provides access to system bus and data bus.

Arbitration, to realize the QFLASH read operation in CACHE mode.

24.2 Main Features

- ÿ Provide bus access FLASH interface
- ÿ Support to directly send SPI commands to FLASH through register configuration
- ÿ Support 24-bit or 32-bit address access to FLASH
- ÿ Support automatic decryption and reading of code and data stored in FLASH

24.3 Functional Description

24.3.1 BUS ACCESS

The address of FLASH in the system starts from 0x08000000, and the address space can be read directly.

24.3.2 Register Access

Through the configuration of registers, SPI commands can be sent directly to FLASH to achieve more flexible access to FLASH, supporting most Sub-FLASH control commands.

24.3.3 Command configuration and startup

The control command code of the Flash is written into the command bit of the register FLASH_CMD, and the command bit of the register is configured according to the command format.

For other bits, for example, if there is an address bit in the command, set the address bit to 1. If there is data in the command, set the DAT bit with

Body reference register description.

After the command configuration is completed, set the command_b bit of the CMD_START register to 1, and the controller starts to send the command to the FLASH.

24.3.3.1 Communication mode and configuration

First send a command to FLASH to configure FLASH as QIO or QPI mode, then set the QIOM or QPIM of the FLASH controller

Set the bit to 1 to use QIO mode or QPI mode to communicate with FLASH.

24.3.3.2 Data Cache

When reading and writing data to FLASH through registers, use the storage space at the address of 0x4000 0000 ~ 0x4000 03ff as the data.

According to the data cache, this address space is also used as the data cache used by the RSA encryption and decryption module. When reading data, the controller will read from FLASH

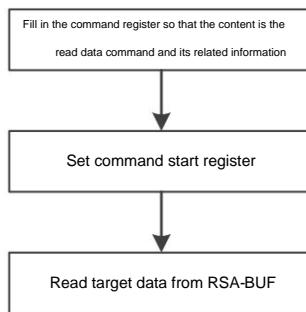
The output data is stored sequentially in the address space starting from 0x4000 0000 for caching. When writing data to FLASH, it is also from 0x4000 0000 Start to read data and send to flash.

24.3.3.3 READ OPERATION

Each read operation can read up to 256 bytes of data, and can read data from any position in QFlash.

After the read operation command is started, the data is stored in the cache without waiting or querying.

The operation flow is shown in the figure.



24.3.3.4 Write Operation

The following commands are all write operations:

WRSR: Write Status Register

PP: Page Programming

SE: sector erase

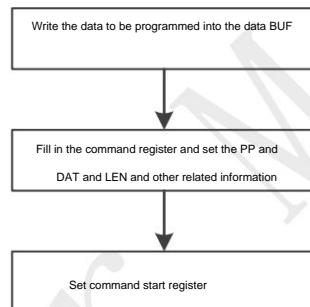
BE: 32K block erase

HBE: 64K block erase

CE: Full Chip Erase

The erase command does not require data, directly configure the command register to the corresponding command, and then start it.

The programming command process is as follows



24.4 Register Description

24.4.1 Register List

Table 201 QFLASH controller register list

offset address	name	abbreviation	describe	Defaults
0X0000_0000	Command information register	CMD_INFO	Command and data words required for QFLASH operation Festival	0X0000_0000
0X0000_0004	Command start register	CMD_START	QFLASH command to start	0X0000_0000
0X0000_0008	Flash control register Flash_CR		QFLASH operating mode control	0X0000_0000
0X0000_000C	Remap register	Remap	Remap option control	0X0000_0000
0X0000_0010	ADDR register	Flash_ADDR	QFLASH operation address	0X0000_0000
0X0000_0014	Decryption control register	DECRYPT_CR	Firmware confidential mode control	0X0000_0000
0X0000_0018	Decryption Status Register	DECRYPT_STA	firmware decryption status;	0X0000_0000
0X0000_0200 ... 0X0000_0600	Read and write data buffer	RSA_BUF	Used to cache QFLASH read and write data or RSA encryption and decryption data. This memory is QFLASH controller and RSA module are shared, Therefore, QFLASH read and write and RSA operations Cannot be used at the same time, it needs to be implemented in the driver Increase mutual exclusion protection	0X0000_0000

24.4.2 Command Information Register

Table 202 QFLASH command information register

bit access	Instructions	reset value

[31]	RW 1: Indicates that CMD_START carries the Address, and the CMD_START register contains the lower 20 bits of the Address Bit, the high 4 of the Address is fixed to 0, a total of 24 bits	1'b0
[30]	RW 1: Indicates that the CRM in the CMD_START register carries the content	1'b0
[29]	RW 1: Indicates that CMD_INFO carries the Dummy period	1'b0
[28:26]	RW indicates how many Dummy cycles CMD_INFO carries after its content +1	3'b0
[25:16]	RW indicates how many bytes of data CMD_INFO carries after its content +1	10'b0
[15]	RW 1: Indicates that CMD_INFO carries data	1'b0
[14]	RW 1: Indicates that the command filled in the current CMD_INFO command field is a read command, and the command includes RDSR, FR, QIOFR, RDPRID, RSR, etc.	1'b0
[13]	RW 1: Indicates that the command filled in the current CMD_INFO command field is the WRSR command	1'b0
[12]	RW 1: Indicates that the command filled in the current CMD_INFO command field is the PP command	1'b0
[11]	RW 1: Indicates that the command filled in the current CMD_INFO command field is the SE command	1'b0
[10]	RW 1: Indicates that the command filled in the current CMD_INFO command field is the BE command	1'b0
[9]	RW 1: Indicates that the command filled in the current CMD_INFO command field is the HBE command	1'b0
[8]	RW 1: Indicates that the command filled in the current CMD_INFO command field is a CE command	1'b0
[7 : 0]	RW CMD_INFO command field, fill in the QFlash command, please refer to the QFlash chip manual.	8'b0

24.4.3 Command Start Register

Table 203 QFLASH command start register

bit access		Instructions	reset value
[31:29] RO		Reserved	3'b0

[28]	RW	When RW is set to 1, it means to start the command. After the command operation is completed, it will be cleared by hardware.	1'b0
[27: 8]	RW	Address[19:0]: If the A flag in the command register is 1, the lower 20 bits of the Address are stored here.	20'b0
[7 : 0]	RW	CRM[7:0]: If the CRM flag in the command register is 1, the CRM content is stored here.	8'b0

24.5 Common Commands of QFLASH

Table 204 QFALSH common commands

command name	Command word	command abbr
Write Enable	06H	WREN
Write Disable	04H	WRDI
Read Status	05H	RDSR
Write Status	01H	WRSR
Fast Read	0BH	FR
Quad IO Fast Read	EBH	QIOFR
Page Program	02H	PP
Sector Erase	20H	SE

For other commands, please refer to the related manuals of QFLASH.

Winner Micro

25 PSRAM interface controller

25.1 Function overview

W800 has a built-in PSRAM controller with SPI/QSPI interface, supports external PSRAM device access, and provides bus-based PSRAM reading

Write and erase operations. The maximum read and write speed is 80MHz.

25.2 Main Features

- ÿ Support read and write access to external PSRAM

- ÿ Configurable as SPI and QSPI

- ÿ SPI/QSPI clock frequency can be configured

- ÿ Support BURST INC mode access

- ÿ Support half sleep mode of PSRAM

25.3 Functional Description

The full name of PSRAM is Pseudo static random access memory, which refers to pseudo-static random access memory. Compared with traditional SRAM

It has the advantages of small package, large capacity and low cost, and is mainly used for data caching in IoT applications. The interface is mostly SPI, QSPI

Wait. The interface pins mainly include clock signal SCK, chip select signal CS and 4 bidirectional data IOs. The PSRAM controller provided by the W800 can

It is accessed by the bus mode of PSRAM supporting SPI/QSPI interface, the maximum working clock rate is 80MHz, and the maximum capacity supports 64Mb.

25.3.1 Pin Description

SCLK: SPI interface clock, the SCLK cycle is set by PSRAM_CTRL[7:4], the minimum frequency division value that can be set is 3, and the default is AHB

Divide by 4 of the clock.

CS: SPI interface chip select signal

SIO0: SPI mode data input, QSPI mode SD[0]

SIO1: SPI mode data output, QSPI mode SD[1]

SIO2: QSPI mode SD[2]

SIO3: QSPI mode SD[3]

25.3.2 Access Mode Settings

The PSRAM controller supports two access modes: SPI and four-wire QSPI to the external PSRAM, and the default is SPI. by setting

PSRAM_CTRL[1] configures the mode of the SPI.

PSRAM_CTRL[1] is 0 by default, that is, SPI mode. At this time, it takes 64 SCLK cycles to complete a write operation and 64 SCLK cycles to complete a read operation.

72 SCLK cycles are required.

If PSRAM works in SPI mode, when writing 1 to PSRAM_CTRL[1], the controller will send command 35H to PSRAM,

When PSRAM_CTRL[1] is read as 1, it means that the command is sent and the PSRAM enters the QPI mode.

16 SCLK cycles, 22 SCLK cycles are required to complete a read operation.

If PSRAM works in QPI mode, when writing 0 to PSRAM_CTRL[1], the controller will send command F5H to PSRAM,

When PSRAM_CTRL[1] is read as 0, it indicates that the command transmission is completed, and the PSRAM enters the SPI mode.

The PSRAM working mode must be set after the initialization operation is completed, but cannot be set at the same time.

25.3.3 PSRAM initialization

Before the first use, after the PSRAM is powered on and stabilized, write 1 to the register PSRAM_CTRL[0] to start the PSRAM reset initialization

operation, that is, send 66H and 99H commands to PSRAM. By default, SPI mode is used to send, that is, 8 SCLK + tCPH + 8

time of SCLK. After initialization, the hardware automatically clears PSRAM_CTRL[0].

The initialization operation will restore the PSRAM to SPI mode.

The recommended initialization process is:

ÿ Write PSRAM_CTRL[0] to 1

(2) Wait until PSRAM_CTRL[0] is automatically cleared

(3) Reset the PSRAM controller by soft reset

ÿ Reset other required parameters of PSRAM_CTRL

25.3.4 Access Methods of PSRAM

The way of reading and writing to PSRAM is the same as that of ordinary SRAM, that is, writing/reading data to the corresponding bus address.

25.3.5 BURST function

By setting PSRAM_CTRL[2], the controller can support the BURST initiated by the AHB bus, that is, when the HBURST on the AHB bus is

At 1/3/5/7, it means that the AHB bus starts a continuous read/write with an address increment. At this time, in order to improve the access speed of PSRAM, control the

After completing the read/write of a word, the processor will not pull CS high, but directly read/write the data of the next word.

The OVERTIMER register is used to set the maximum time for CS to be low, the unit is the number of cycles of HCLK, each time a BURST operation starts,

The internal counter starts counting from 0. When the counter value is greater than the set value, the controller stops automatically after completing the read/write of the current word.

BURST operation, directly change CS to high level, if there is still data on the AHB bus to read/write at this time, it will be treated as a separate WORD

conduct.

The PSRAM controller does not support BURST in the form of WRAP. If WRAP BURST is used to access PSRAM, please

PSRAM_CTRL[2] is set to 0.

25.4 Register Description

25.4.1 Register List

Table 205 PSRAM Controller Register List

offset address name		abbreviation	describe	Defaults
0X0000_0000 Control Register		PSRAM_CTRL	PSRAM Controller Settings	0X0000_0000
0X0000_0004 Timeout Control Register		OVERTIMER	CS Timeout Control	0X0000_0000

25.4.2 Command Information Register

Table 206 PSRAM Control Setting Register

bit access		Instructions	reset value
[31:12] RO		RSV	'b0
[11]	RW HSM	Halsleep mode enabled 1: Enable PSRAM half-sleep mode 0: Clear half sleep mode	1'b0
[10:8]	RW tCPH	the shortest time setting of CS high level, the unit is the number of AHB clock cycles, must be greater than 1, the specific time depends on the Set it according to the instructions in the same psram manual, if you don't know, you can not modify the default value.	3'd6
[7:4]	RW SPI frequency divider setting	It can only be configured to a value of 2 or more, and the written value is a multiple of the frequency division	4'd4
[3]	RO	RSV	
[2]	RW INC_EN	BURST function enable 1: Support BURST function on AHB bus 0: BURST function is not supported	1'b0

[1]	RW QUAD	Write 1 to enable QPI mode of PSRAM, write 0 to enable SPI mode Read this flag to know which mode the current PSRAM is in.	1'b0
[0]	RW PSRAM reset	Write 1 to start the reset operation to PSRAM, and it will be automatically cleared after reset.	1'b0

25.4.3 Timeout Control Register

Table 207 CS Timeout Control Register

bit access		Instructions	reset value
[11:0] RW	Timeout register setting, sets the maximum time for CS to be low, for BURST mode		12'd0

26 ADCs

26.1 ADC Function Overview

The acquisition module based on Sigma-Delta ADC completes the acquisition of up to 4 channels of analog signals. The sampling rate is controlled by an external input clock.

It can collect input voltage and chip temperature, and supports input calibration and temperature compensation calibration.

26.2 ADC Main Features

- ÿ Support up to 4 channels of data acquisition
- ÿ Support DMA module for data buffering;
- ÿ Support interrupt interaction mode;
- ÿ Support the function of comparing the collected data with the input data
- ÿ The highest sampling frequency is 1KHz;
- ÿ Support single-ended input mode and differential input mode;

26.3 ADC Function Description

26.3.1 Module Basic Structure

The ADC module is a PGA+SDADC structure. After the input signal is pre-amplified by the PGA, it is input to the SDADC for processing. The SDADC design

For 16bit ADC. It should be noted that the on-chip ADC is powered by a 2.5V LDO. In order to protect the internal circuit, the input signal must be fully

Foot Vinmax < 2.5V, other restrictions, see section 26.3.8.

26.3.2 Channel selection

Register 0x04[11:8] provides the channel selection function. The ADC of W800 provides 4-channel signal acquisition function. by setting the register

0x04[11:8] can select any of the 4 channels for single-ended signal data acquisition, or select 2 pre-paired differential pairs in the 4 channels

Differential signal data acquisition by channel. For specific channel selection, please refer to the register description.

In addition to 4 channels, the ADC module also provides temperature detection, voltage detection and offset calibration functions. corresponding to the channel selection

4'b1100, 4'b1101, 4'b1110.

If you need to use the corresponding function, configure the channel selection as the corresponding channel, and then follow the process of the above instructions.

26.3.3 Data Comparison

The ADC module provides data comparison function, the switch of this function can be set by BIT[5] of configuration register 0x10.

The user can configure the value to be compared through BIT[17:0] of result register 0x18. This value needs to be converted to a 18bit signed number, the most

The high bit is the sign bit. BIT[6] of ADC configuration register 0x10 can set the direction of data comparison, when this bit is 0, if ADC

If the collected value is greater than or equal to Comp_data, INT_CMP will be set to 1 and an interrupt will be generated; when this bit is 1, only

When the value collected by the ADC is less than Comp_data, INT_CMP will be set to 1 and an interrupt will be generated.

26.3.4 PGA Gain Adjustment Instructions

BIT[8:4] of register address 0X08, 5-bit gain control signal, among which

0X08[8:7]=GAIN_CTRL_PGA<4:3> (corresponding to GAIN2)

0X08[6:4]=GAIN_CTRL_PGA<2:0> (corresponding to GAIN1, 110 and 111 are not used)

PGA overall gain GAIN_PGA=GAIN1*GAIN2

PGA gain configuration table (represented by magnification)

[8:7] [6:4]	00	01	10	11
000	1	2	3	4
001	16	32	48	64
010	32	64	96	128
011	64	128	192	256
100	128	256	384	512
101	256	512	768	1024

According to the simulation results, for the same magnification, it is recommended to use the configuration with the larger magnification of GAIN2.

26.3.5 Single-Ended Mode VCMIN Adjustment Description

The internal VCMIN voltage of the PGA, in single-ended mode, is used as the negative terminal of the op amp input.

The purpose of adjusting VCMIN is to adapt to various signal sources with different bias points and prevent PGA output from saturating at some too high or too low bias points.

and.

BIT[16:11] of register address 0X40000D20, default 100000, ie VCMIN default=1.311V, step \geq 39.1mV

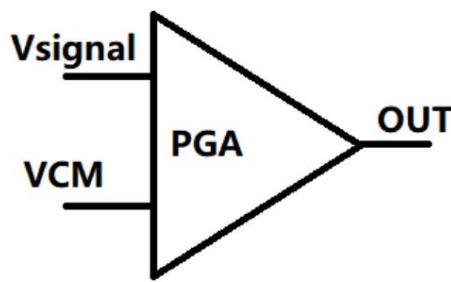


Figure 37 Principle of single-ended working mode

26.3.6 Bypass Mode

Bypass_pga register address 0X08 BIT[3], the default is 0, not bypassed, the PGA works normally; if it is 1, the PGA is bypassed at this time,

The external signal source is directly connected to the SDADC input, which is mainly used for internal module testing, and no bypass is required in normal use.

Bypass_ref register address 0X08 BIT[2], the default is 0, no bypass, SDADC works normally; if it is 1, then the SDADC

The partial ref is bypassed, and an external reference voltage needs to be added through the test pin. It is mainly used for internal module testing, and bypass is not required in normal use.

26.3.7 Supplementary description of SDADC output code

SDADC output result is stored in ADC_RESULT register, register address 0X00 BIT[17:0].

The output code of SDADC itself is an unsigned number, because the number processes the output code of SDADC (the highest bit is inverted), it becomes signed

number, and the 0 code of the SDADC output code is not fixed (and not exactly 10 0000 0000 0000 0000), so the register is directly used.

The value of the device for subsequent calculations is wrong.

When calculating the output result of SDADC, the value of register ADC_RESULT[17:0] (the read value is a hexadecimal number) requires software to

Convert back to an unsigned number (the highest bit ADC_RESULT[17] of the binary number is inverted), and then perform subsequent calculations.

The SDADC designed this time is a 16bit ADC (the design effective number of bits is 16bit), and the register data is 18bit, which is increased by a digital filter.

16bit number is converted to 18bit number (18bit number \geq 16bit number*4), so the effective number of digits is the upper 16bit (ie

ADC_RESULT[17:2]).

After converting the read value back to an unsigned number, the calculation involving the LSB also requires software to convert the 18-bit data into 16-bit data, that is, discard the most

The lower two bits of data (ADC_RESULT[1:0]), only the upper 16bits are reserved (returned to unsigned ADC_RESULT[17:2]), and then 16bits are used

The data is used for subsequent calculation. At this time, the simulation result of LSB is approximately equal to 68.5uV (the actual LSB of the chip needs to be tested to determine).

26.3.8 Input signal voltage range

Because the NTO version PGA and SDADC use 2.5V LDO power supply, in order to prevent internal circuit saturation, the input signal voltage range is limited system.

Assuming that the input differential signals are Vinp and Vinn (in single-ended mode, the signal Vinn is VCMIN), then the input differential voltage Vdiff=Vinp

Vinn, input common mode voltage Vcm=(Vinp+Vinn)/2. PGA first stage gain GAIN1, second stage gain GAIN2, PGA overall gain

GAIN_PGA=GAIN1*GAIN2 (see Section 2.2.2 for details).

The input signal must meet the following constraints at the same time:

0V<Vinp<2.5V, 0V<Vinn<2.5V;

0V<Vcm-(Vdiff*GAIN1)/2;

Vcm+(Vdiff*GAIN1)/2<2.5V;

0V<1.2V-(Vdiff*GAIN_PGA)/2;

1.2V+(Vdiff*GAIN_PGA)/2<2.5V;

If the approximate range of the input signal is known, it can be directly substituted into the above formula to calculate and select the available gain setting.

For unknown input signal, you can first use the x1 gain setting to determine the approximate range, and then substitute it into the above formula to calculate and select the available gain setting.

26.4 Register Configuration for Typical Use Cases

The following configuration words are all hexadecimal numbers, and the high-order 0 has been omitted.

26.4.1 Offset measurement

1. Set the working state of the module, set register 0X04 to 0xE03, and enable channel selection (offset detection)/SDADC_CHOP to enable

enable/SDADC enable/LDO enable

2. Set module register 0X08=0x3, PGA is x1 gain configuration/PGA_CHOP enable/PGA enable. Among them, the gain configuration

0X08 BIT[8:4] should be modified according to the requirements of the test (that is, what gain configuration is used for the subsequent test, offset measurement

also use the same gain configuration)

3. Set the clock state 0X40000E14 BIT[15:8]=0x28, and confirm that the SDADC clock is configured as 1MHz. The default value after the chip is powered on

That is 0x28, if there is no change, this step can be omitted

4. Wait for 2ms, read the output result of ADC_RESULT register, refer to Section 26.3.7, need software to convert back to unsigned number (most

The high bit ADC_RESULT[17] is inverted), and then the lowest two bits are discarded, and only the high 16bit data is retained. Repeat the reading for a total of 10 times, the reading interval is 2ms,

Calculate the average number of codes and record it as code_offset

26.4.2 Differential Input Mode

1. First measure the offset according to Section 26.4.1 to get code_offset

2. Set the module register 0X04=0x803, channel selection (positive channel 0, negative channel 1)/SDADC_CHOP enable

/SDADC enable/LDO enable. The channel selects 0X04[11:8], which can be configured as the differential input of other channels, refer to 26.3.2

Festival:

3. Set module register 0X08=0x3, PGA is x1 gain configuration/PGA_CHOP enable/PGA enable. Among them, the gain configuration

0X08[8:4] Select the appropriate configuration according to the input signal voltage range, refer to Sections 26.3.4 and 26.3.8

4. Set the clock state 0X40000E14[15:8]=0x28, and confirm that the SDADC clock is configured as 1MHz. After the chip is powered on, the default value is

28. If there is no change, this step can be omitted

5. Wait 2ms, read the output result of ADC_RESULT register, register address 0X00[17:0]. Refer to Section 26.3.7, required

The software first converts back to an unsigned number (the highest bit ADC_RESULT[17] is inverted), and then discards the lowest two bits, and only retains the high 16bit data. 6.

Repeat the reading for a total of 10 times, the reading interval is 2ms, and the average value of the code number is calculated as code_out

7. Differential input signal $V_{in,diff} = (code_out - code_offset) * LSB / GAIN_PGA$, where LSB is approximately equal to

68.5uV

26.4.3 SINGLE-ENDED INPUT MODE

1. First measure the offset according to Section 26.4.1 to get code_offset;

2. Set module register 0X04=0x3, channel selection (positive channel 0, negative VCMIN)/SDADC_CHOP enable/SDADC

Enable/LDO enable. The channel selects 0X04[11:8], which can be configured as the single-ended input of other channels, refer to Section 26.3.2

Set module register 0X408=0x3, PGA is x1 gain configuration/PGA_CHOP enable/PGA enable. Among them, VCMIN configuration

0X40000D20[16:11] and gain configuration 0X08[8:4], select the appropriate configuration according to the input signal voltage range, refer to 26.3.4,

Sections 26.3.5 and 26.3.8

3. Set the clock state 0X40000E14[15:8]=0x28, and confirm that the SDADC clock is configured as 1MHz. After the chip is powered on, the default value is

28. If there is no change, this step can be omitted;

4. Wait 2ms, read the output result of ADC_RESULT register, register address 0X00[17:0]. Refer to Section 26.3.7, required

The software first converts back to an unsigned number (the highest bit ADC_RESULT[17] is inverted), and then discards the lowest two bits, and only retains the high 16bit data. repeat

Read a total of 10 times, the reading interval is 2ms, and the average value of the code number is calculated as code_out;

5. Single-ended mode input signal $V_{in,single} = (code_out - code_offset) * LSB / GAIN_PGA + VCMIN$, where LSB according to

The simulation result is approximately equal to 68.5uV, and the default value of VCMIN is 1.311V;

26.4.4 Temperature detection mode

1. Set the module register 0X04=0xC03, channel selection (temperature detection)/SDADC_CHOP enable/SDADC enable/LDO enable

can;

2. Set module register 0X08=0x183, PGA is x4 gain configuration/PGA_CHOP enable/PGA enable;

3. Set the clock state 0X40000E14[15:8]=0x28, and confirm that the SDADC clock is configured as 1MHz. After the chip is powered on, the default value is

28. If there is no change, this step can be omitted;

4. Set the working state of tempsensor module 0X0C=0x1, TempSensor enable/cal_offset_temp12=0,

TempSensor is configured for x2 gain;

5. Wait for 2ms, read the output result of ADC_RESULT register, refer to Section 26.3.7, need software to convert back to unsigned number (most

The high-order ADC_RESULT[17] is inverted), recorded as output code code_out1;

6. Set the working state of tempsensor module 0X0C=0x3, TempSensor enable/cal_offset_temp12=1,

TempSensor is configured for x2 gain;

7. Wait 2ms, read the output result of ADC_RESULT register, refer to Section 26.3.7, need software to convert back to unsigned number (most

The high-order ADC_RESULT[17] is inverted), recorded as the output code code_out2;

8. Repeat steps 4 to 7 for a total of 10 times to calculate the average values of code_out1avg and code_out2avg of code_out1 and code_out2 respectively.

code_out2avg;

9. Calculate Vtemp=(code_out1avg-code_out2avg)/16 from the above steps, and TempSensor output voltage=Vtemp

Code number* (LSB/4), where LSB is approximately equal to 68.5uV according to the simulation result (this LSB is the simulation result according to 16bit, the temperature detection above

In steps 5 and 7, the reading is 18bit data, 18bit number ÷ 16bit number * 4, then the least significant bit corresponding to 18bit ÷ LSB/4)

10. According to the test results, the fitting formula has been sorted out, which can directly estimate the chip temperature

Vtemp code=(15.548*Chip temperature)+4444.1

Substitute the Vtemp calculated in step 9 into the above formula to obtain the die temperature. (Note that the fitting formula is in decimal)

Also, add an optional measure to improve accuracy:

For different chips, assuming that the coefficient k in the fitting formula $y=kx+b$ is constant (both are 15.548), the coefficient b of different chips is different to some extent.

According to the above instructions, first calculate the Vtemp code at a known ambient temperature (eg room temperature 25°C), and substitute it into the Vtemp code

$=15.548*(\text{known ambient temperature}+11.8)+b$, get the coefficient b of different chips, save it in rom or flash, replace the above formula

4444.1. Then, the temperature is calculated according to the coefficient of the new formula, which can improve the temperature detection accuracy of different chips to a certain extent.

26.4.5 Voltage Detection Mode

1. First measure the offset according to Section 26.4.1 to get code_offset;
2. Set module register 0X04=0xD03, channel selection (voltage detection)/SDADC_CHOP enable/SDADC enable/LDO enable

can

3. Set module register 0X08=0x83, PGA is x2 gain configuration/PGA_CHOP enable/PGA enable.
4. Set the clock state 0X40000E14[15:8]=28, and confirm that the SDADC clock is configured as 1MHz. After the chip is powered on, the default value is 28.

If there is no change, this step can be omitted

5. Wait for 2ms, read the output result of ADC_RESULT register, refer to Section 26.3.7, need software to convert back to unsigned number (most

The high bit ADC_RESULT[17] is inverted), and then the lowest two bits are discarded, and only the high 16bit data is retained. Repeat the reading for a total of 10 times, the reading interval is 2ms,

Calculate the average number of codes and record it as code_out;

6. Chip power supply voltage $V_{power} = (code_out - code_offset) * LSB / 2 + 1.2V$, where LSB is approximately equal to the simulation result

68.5uV

26.5 ADC Register Descriptions

26.5.1 Register List

Table 208 ADC register list

offset address name		abbreviation	describe	reset value
0X0000	ADC result register	ADC_RESULT	Store ADC acquisition value and data comparison value	0X0000_0000
0X0004	ADC analog register	ADC_ANA_CTRL	Configure ADC related functions	0X0000_0004
0X0008	PGA Configuration Register	PGA_CTRL	Configure PGA related functions;	0x0000_0000
0X000c	TempSensor configuration register <small>device</small>	TEMP_CTRL	Configure temperature sensor related functions; 0x0000_0000	
0x0010	ADC Configuration Register	ADC_CTRL	Configure ADC module function;	0x0050_0500
0x0014	ADC Interrupt Register	ADC_INT_STATUS	ADC Module Interrupt Status Register; 0x0000_0000	
0x0018	Compare Value Register	CMP_VALUE	Compare threshold settings	0x0000_0000

26.5.2 ADC Result Register

Table 209 ADC result register

bit access		Instructions	reset value
[17:0] RW	ADC conversion	result value, valid bit width is 18bits. Signed number, the most significant bit is the sign bit.	1'b0

26.5.3 ADC ANALOG CONFIGURATION REGISTER

Table 210 ADC configuration registers

Bit Access	Operating Instructions		reset value
[11 : 8] RW	<p>ADC working channel selection signal:</p> <p>4'b0000: AIN0 channel works. In DMA mode, the corresponding DMA channel 0 works</p> <p>4'b0001: AIN1 channel works. In DMA mode, the corresponding DMA channel 1 works</p> <p>4'b0010: AIN2 channel works. In DMA mode, the corresponding DMA channel 2 works</p> <p>4'b0011: AIN3 channel works. In DMA mode, the corresponding DMA channel 3 works</p> <p>4'b0100: RSV</p> <p>4'b0101: RSV</p> <p>4'b0110: RSV</p> <p>4'b0111: RSV</p> <p>4'b1000: AIN0/AIN1 differential signal input. In DMA mode, the corresponding DMA channel 0 works</p> <p>4'b1001: AIN2/AIN3 differential signal input. In DMA mode, the corresponding DMA channel 2 works</p> <p>4'b1010: RSV</p> <p>4'b1011: RSV</p> <p>4'b1100: Temperature sensor input, corresponding to DMA channel 2</p> <p>4'b1101: Voltage detection module input, corresponding to DMA channel 3</p> <p>4'b1110: offset detection input;</p> <p>4'b1111: RSV</p>		4'b1000
[7]	RO	RSV	1'b0
[6:5] RW	chop_enr		2'b00

		LDO chopper signal PD signal	
[4]	RW	<p>Chop_ens</p> <p>sdadc chopped signal PD signal</p> <p>0: enable</p> <p>1: Disable</p>	1'b0
[3]	RO	RSV	1'b0
[2]	RW	<p>Pd_sdadc</p> <p>sdadc analog module power down enable</p> <p>1: Power down</p> <p>0: work</p>	1'b1
[1]	RW	<p>Rstn_sdadc</p> <p>Reset signal of the digital logic part of the analog module</p> <p>0: reset</p> <p>1: normal work</p>	1'b0
[0]	RW	<p>en_ldo_sdadc</p> <p>ADC LDO enable</p> <p>0: Power down 1: Working</p>	1'b0

26.5.4 PGA CONFIGURATION REGISTERS

Table 211 PGA configuration registers

Bit Access	Operating Instructions		reset value
[8:4] RW		<p>Gain_ctrl_pga</p> <p>PGA gain configuration;</p> <p>BIT[8:7] configure GAIN2, BIT[6:4] configure GAIN1, please refer to Section26.3.4 for the specific gain table</p>	5'd0
[3]	RW	<p>Bypass_pga</p> <p>pga bypass signal</p> <p>1: Bypass pga</p> <p>0: do not bypass</p>	1'b0
[2]	RW	<p>Bypass_ref</p> <p>Internal voltage reference bypass signal</p> <p>1: Bypass the internal reference voltage</p> <p>0: do not bypass</p>	1'b0
[1]	RW	<p>Chop_enp</p> <p>PGA chop enable signal</p> <p>1: enable</p> <p>0: Disable</p>	1'b0
[0]	RW	<p>En_pga</p> <p>Pga enable signal</p> <p>1: enable</p> <p>0: Disable</p>	1'b0

26.5.5 TEMP CONFIGURATION REGISTER

Table 212 Temperature Sensor Configuration Register

Bit Access	Operating Instructions		reset value
[5:4] RW		Gain_temp temp gain control Coding Gain 00 -- 2 01 -- 4 10 -- 6 11 -- 8	2'd0
[3:2] RO		RSV	2'b0
[1]	RW	Cal_offset_temp12 TEMPSEN offset calibration function	1'b0
[0]	RW	ON_TEMP 1: temp enable 0: temp is not enabled	1'b0

26.5.6 ADC Function Configuration Register

Table 213 Temperature Sensor Configuration Register

Bit Access	Operating Instructions		reset value
[29:20] RW		ana_swth_time After the software switches the data channel, the time required for the analog circuit to remain stable, the default is 80 pclk, or 2us	10'h050

[19:18] RO		RSV	2'b0
[17:8] RW		<p>ana_init_time</p> <p>After the software starts adc_start, the time required for the analog circuit to remain stable, the default is 80 pclk, that is, 2us</p>	10'h050
[7]	RO	RSV	1'b0
[6]	RW	<p>Cmp_pol</p> <p>0: interrupt when adc_result >= cmp_value</p> <p>1: interrupt when adc_result < cmp_value</p>	1'b0
[5]	RW	<p>Cmp_int_ena</p> <p>1: Compare interrupt enable</p> <p>0: Compare interrupt disabled</p>	1'b0
[4]	RW	<p>Adc_cmp_enable</p> <p>1: adc compare function is enabled</p> <p>0: adc comparison function is disabled</p>	1'b0
[3:2]	RO	Reserved	2'b0
[1]	RW	<p>Adc_int_ena</p> <p>1: ADC data conversion interrupt enable</p> <p>0: ADC data conversion interrupt disabled</p>	1'b0
[0]	RW	<p>Adc_dma_enable</p> <p>1: dma enable</p> <p>0: dma is not enabled</p>	1'b0

26.5.7 ADC Interrupt Status Register

Table 214 ADC Interrupt Status Register

Bit Access	Operating Instructions		reset value
[1]	RW	Cmp_int Compare interrupt flag bit, set by hardware, cleared by software by writing 1	1'b0
[0]	RW	Adc_int Data conversion complete interrupt, hardware set, software write 1 to clear	1'b0

26.5.8 Compare Threshold Register

Table 215 Compare Threshold Register

Bit Access	Operating Instructions		reset value
[17:0] R/W		Cmp_value value to compare	18'h00000

27 Touch Sensor

27.1 Overview of Module Functions

The basic functions of the module are as follows:

↳ Supports up to 16 Touch Sensor scans;

↳ Record the scanning results of each Touch Sensor;

↳ Reporting scan results through interrupts;

27.2 Function Instructions

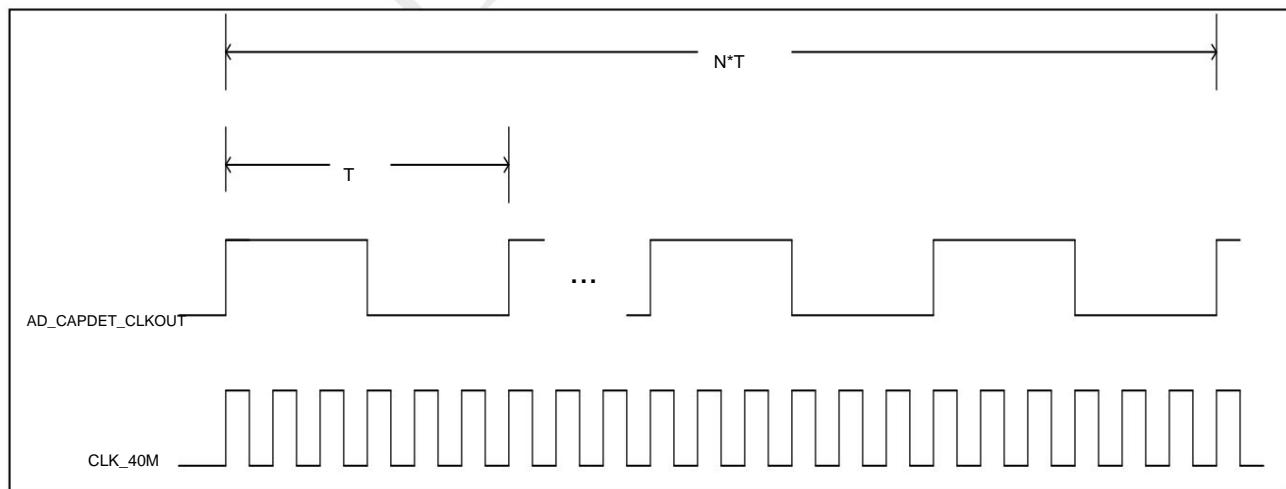
The touch button module is integrated in the W800 system. The basic principle is that when the button is touched, the capacitance value of the circuit on the button will change, so that the

Affects the output clock frequency in the module. By detecting the frequency of the output clock of the module, it can be judged whether the capacitance value has changed, so as to judge

The state of the button.

The basic function of this digital module is to scan the state of each touch key at regular intervals, count and record within the set time window.

the state of each button. If it exceeds the set threshold, it is judged that the button is touched, and it is reported to the MCU system through an interrupt.



27.2.1 Basic Workflow

1. Set the Touch_CR register, configure the scan cycle, scan window, and select the IO to be scanned. In Touch_CR

scan_period is the interval period of each scan, the unit is 16ms. That is, if the register is set to 10, every 160ms will be

Scan 16 touch keys. CAPDET_CNT is the window counted when scanning each IO state, which is N in the above figure. It should be noted that for

To avoid the jitter caused by switching the channel, the counting will start at the beginning of the third pulse after each switching of the scan IO, so if this register is set
is N, the actual count window is N-2.

2. Set the count threshold corresponding to each Touch IO. If the result of the scan exceeds the base + threshold, the button can be considered to be touched;

3. Enable Touch_CR[0], the module starts to work.

4. After the touch key module is enabled, the hardware will scan the selected IO one by one, and store the count value of each IO as the base.

Then scan the selected IO successively every scan_period, and store the currently scanned value. After all IO scans are completed, the

The count value of each IO is compared with the radix, if the current value > radix+threshold, the button is considered to be touched, in register 0x44

The corresponding bit in PAD_STATUS will be set to 1 and reported to the system through an interrupt. PAD_STATUS Write 1 to clear 0.

27.3 Register List:

Table 216 Touch Sensor Controller Register List

offset address name		abbreviation	describe	Defaults
0X0000_0000	Control Register	Touch_CR	Touch Sensor Controller Settings	0X0000_0000

0X0000_0004			Threshold control and count value of each touch key 0X0000_0000	
~	One-way control of touch buttons	Touch_Sensor x		
0X0000_0040				
0x0000_0044	Interrupt Controller	Int_Source	interrupt controller	0x0000_0000

27.3.1 Touch Sensor Control Register

Table 217 Touch Sensor Control Setting Register

bit access		Instructions	reset value
[31:26] RW	Scan_period	<p>Scan period, the unit is 16ms; for example, if scan_period is set to 6'd10, then scan every 160ms.</p> <p>key state;</p>	6'd10
[25:20] RW	CAPDET_CNT	<p>Select the number of CAPDET output pulses as the count window.</p> <p>Note: In order to avoid jitter caused by switching channels, counting will not start until the third pulse, so this register is set to N-2.</p> <p>If it is set to N, the actual count window is N-2. For example, if it is set to 6'd20, the cycle of 18 CAPDET pulses is the count window.</p>	6'd20
[19:4] RW	Touch_Sensor	<p>key selection; scan the touch key state of the corresponding bit.</p> <p>0x0000: do not scan;</p> <p>0x0001: scan the first key;</p> <p>0x0002: scan the second key;</p>	16'd0

		0x0003: scan the first and second keys; ... 0xFFFF: scan all 16 keys;	
[3:1]	RW RSV		3'd0
[0]	RW Touch key controller enable [0] 1: Enable touch key scan; 0: Disable touch key scanning;		1'b0

27.3.2 Touch key single control register

Table 218 Touch key single channel setting register

bit access		Instructions	reset value
[22:8] RO		Touch Senor 1 count value	14'd0
[7]	RO	RSV	1'b0
[6:0]	RW	Touch Senor 1 Threshold	7'd50

27.3.3 Interrupt Control Register

Table 219 Touch key interrupt control register

bit access		Instructions	reset value
[31:16]	RW INT_EN	When the corresponding bit is 1, it means that the corresponding IO is triggered and an interrupt will be generated; When the corresponding bit is 0, it means that the corresponding IO is triggered and will not generate an interrupt;	16'd0

[15:0]	RW PAD_STATUS/INT_SOURCE	<p>When the bit is 1, it means that the corresponding PAD is triggered;</p> <p>When the bit is 0, it means that the corresponding PAD is not triggered;</p> <p>write 1 clear 0</p>	16'd0

28 W800 Security Architecture Design

28.1 Functional overview

XT804 expresses safety features of bus access via HPROT signal

	0	1
HPROT[3]	uncacheable	cacheable
HPROT[2]	un-security	security
HPROT[1]	User	super
HPROT[0]	Code	data

W800's support for security architecture is divided into security access control for sram and access control for peripherals.

28.1.1 SRAM Secure Access Controller (SASC)

Level 1 bus SRAM, Level 2 bus SRAM and FLASH each have a security access controller, each SASC controls the storage space have the following safety features

- ÿ 8 configurable security zones

- ÿ Each security area can be configured to a minimum of 4 bytes

- ÿ Security-super provides unrestricted access to all areas

- ÿ If the access of the CPU through the AHB bus is denied, it will generate a wrong response signal (HRESP=1), from

An access exception is thrown instead.

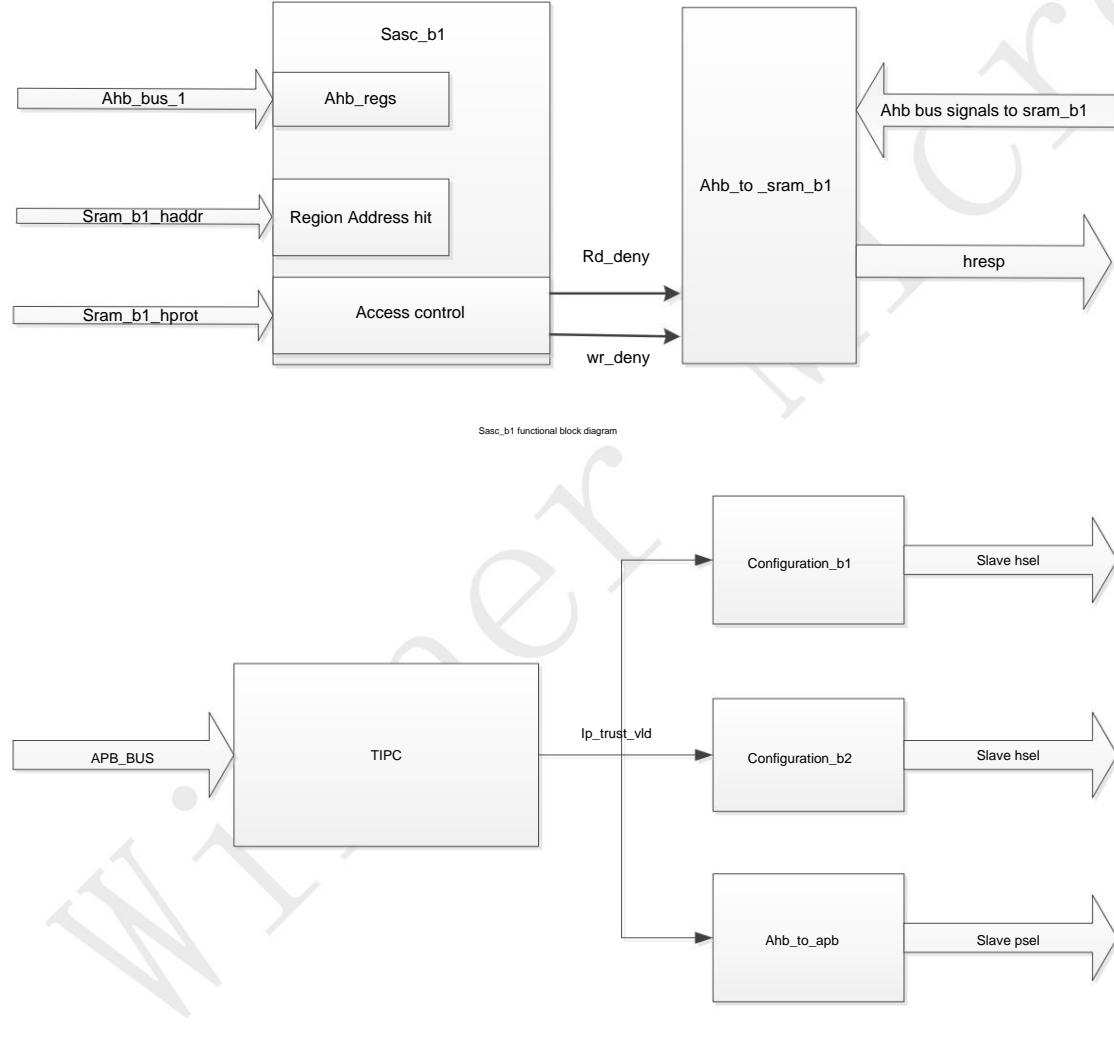
- ÿ No exception will be generated if the access of other master devices such as DMA on the bus is denied.

28.1.2 Trusted IP Controller (TIPC)

Hanging on the APB bus, it is used to configure the trust authority of all peripherals including the primary bus, the secondary bus and the APB bus. If the peripheral is configured as a trusted device (`ip_trust_vld=1`), then only access on the bus is security (HPROT[2]=1 or PPROT[2]=1)

Only then can the peripherals be accessed normally, otherwise the read and write will be rejected.

28.2 Security Architecture Block Diagram



28.3 Register Description

28.3.1 SASC Register List

name offset	address	bit threshold	illustrate	reset value
CAR	0x0	[31:16]	Reserved unused	--
		[15:14]	sram region7 configuration properties, same as region0 2'b00	
		[13:12]	sram region6 configuration properties, same as region0 2'b00	
		[11:10]	sram region5 configuration properties, same as region0 2'b00	
		[9:8]	sram region4 configuration properties, same as region0 2'b00	
		[7:6]	sram region3 configuration properties, same as region0 2'b00	
		[5:4]	sram region2 configuration properties, same as region0 2'b00	
		[3:2]	sram region1 configuration properties, same as region0 2'b00	
		[1:0]	sram region0 configuration properties 00: un-security-user 01: un-security-super 10: security-user 11: security-super	2'b00
CR	0x4	31:8	unused reserve	--
		7	region7 attribute control register, same as region0	
		6	region6 attribute control register, same as region0	
		5	region5 attribute control register, same as region0	
		4	region4 attribute control register, same as region0	
		3	region3 attribute control register, same as region0	

		2	region2 attribute control register, same as region0	
		1	region1 attribute control register, same as region0	
		0	region0 attribute control register 0: Attribute invalid 1: Attribute valid	
	AP0 0x8	[31:16]	Reserved unused	
		[15:14]	region7 AP configuration for un-security user	
		[13:12]	region6 AP configuration for un-security user	
		[11:10]	region5 AP configuration for un-security user	
		[9:8]	region4 AP configuration for un-security user	
		[7:6]	region3 AP configuration for un-security user	
		[5:4]	region2 AP configuration for un-security user	
		[3:2]	region1 AP configuration for un-security user	
		[1:0]	region0 AP configuration for un-security user 00: R/W 01: RO 10: WO 11: No	

			Access	
CD0	0xC	[31:16]	Reserved unused	
		[15:14]	region7 CD configuration for un-security user	
		[13:12]	region6 CD configuration for un-security user	
		[11:10]	region5 CD configuration for un-security user	
		[9:8]	region4 CD configuration for un-security user	
		[7:6]	region3 CD configuration for un-security user	
		[5:4]	region2 CD configuration for un-security user	
		[3:2]	region1 CD configuration for un-security user	
		[1:0]	region0 CD configuration, for un-security user 00: Date Access , Opcode Feche 01: Data Access 10: Opcode Feche 11: Data, Opcode all deny	

AP1	0x10	[31:16]	Reserved unused	
		[15:14]	region7 AP configuration for un-security super	
		[13:12]	region6 AP configuration for un-security super	
		[11:10]	region5 AP configuration for un-security super	
		[9:8]	region4 AP configuration for un-security super	
		[7:6]	region3 AP configuration for un-security super	
		[5:4]	region2 AP configuration for un-security super	
		[3:2]	region1 AP configuration for un-security super	
		[1:0]	region0 AP configuration for un-security super 00: R/W 01: RO 10: WO 11: No Access	
CD1	0x14	[31:16]	Reserved unused	
		[15:14]	region7 CD configuration for un-security super	

		[13:12] region6 CD configuration for un-security super	
		[11:10] region5 CD configuration for un-security super	
		[9:8] region4 CD configuration for un-security super	
		[7:6] region3 CD configuration for un-security super	
		[5:4] region2 CD configuration for un-security super	
		[3:2] region1 CD configuration for un-security super	
		[1:0] region0 CD configuration, for un-security super 00: Date Access , Opcode Feche 01: Data Access 10: Opcode Feche 11: Data, Opcode all deny	
AP2	0x18	[31:16] Reserved unused	
		[15:14] region7 AP configuration for security-user	
		[13:12] region6 AP configuration for security-user	
		[11:10] region5 AP configuration for security-user	

		[9:8]	region4 AP configuration for security-user	
		[7:6]	region3 AP configuration for security-user	
		[5:4]	region2 AP configuration for security-user	
		[3:2]	region1 AP configuration for security-user	
		[1:0]	region0 AP configuration for security-user 00: R/W 01: RO 10: WO 11: No Access	
CD2	0x1C	[31:16]	Reserved unused	
		[15:14]	region7 CD configuration for security-user	
		[13:12]	region6 CD configuration for security-user	
		[11:10]	region5 CD configuration for security-user	
		[9:8]	region4 CD configuration for security-user	
		[7:6]	region3 CD configuration for security-user	
		[5:4]	region2 CD configuration, for security-user	
		[3:2]	region1 CD configuration, for security-user	
		[1:0]	region0 CD configuration, for security-user 00: Date Access , Opcode Feche 01: Data Access 10: Opcode Feche 11: Data, Opcode all deny	
REGION0 0x20		31:n+1	reserved	0
		n:8	BAddr0, region0 base address, n and sram are larger	

			little related, sram=32kB, n=20 sram=64kB, n=21	
		7:5 reserved		3'b000
		4:0	RSize, region0 size 00101: 4B 00110: 8B ... 10001: 16KB 10010: 32KB ...	
		31:n+1 reserved		0
REGION1 0x24		n:8	Baddr1, region1 base address	
		7:5 reserved		3'b000
		4:0	region1 size	
		31:n+1 reserved		0
REGION2 0x28		n:8	region2 base address	
		7:5 reserved		3'b000
		4:0	region2 size	
		31:n+1 reserved		0
REGION3 0x2C		n:8	region3 base address	
		7:5 reserved		3'b000
		31:n+1 reserved		0

		4:0	region3 size	
REGION4 0x30		31:n+1 reserved		0
		n:8	region4 base address	
		7:5 reserved		3'b000
		4:0	region4 size	
REGION5 0x34		31:n+1 reserved		0
		n:8	region5 base address	
		7:5 reserved		3'b000
		4:0	region5 size	
REGION6 0x38		31:n+1 reserved		0
		n:8	region6 base address	
		7:5 reserved		3'b000
		4:0	region6 size	
REGION7 0x3C		31:n+1 reserved		0
		n:8	region7 base address	
		7:5 reserved		3'b000
		4:0	region7 size	

28.3.2 TIPC register

name offset	address	bit threshold	illustrate	reset value
ip_trust_vld0 0x0		31:18 unused		0
		17	BT modem trusted attributes 1: Trusted 0: Untrustworthy	0
		16	I2S Trusted Properties	0
		15	PWM Trusted Properties	0
		14	LCD driver trusted attributes	0
		13	RF controller trusted attributes	0
		12	timer trusted attribute	0
		11	watch dog trusted attributes	0
		10	PORTB Trusted Properties	0
		9	PORTA Trusted Properties	0
		8	UART5 Trusted Properties	0
		7	UART4 Trusted Properties	0
		6	UART3 Trusted Properties	0
		5	UART2 Trusted Properties	0
		4	UART1 Trusted Properties	0
		3	UART0 Trusted Properties	0

		2	SPI MASTER Trusted Properties	0
		1	SAR ADC Trusted Properties	0
		0	I2C Trusted Properties	0
ip_trust_vld1 0x4		31:18 unused		0
		17	RF BIST Trusted Attribute Configuration 1: Trusted 0: Untrustworthy	0
		16	SDIO Wrapper Trusted Properties	0
		15	SPI_HS Trusted Properties	0
		14	SDIO Trusted Properties	0
		13	unused	0
		12	SEC Trusted Properties	0
		11	MAC Trusted Properties	0
		10	BBP Trusted Properties	0
		9	MMU Trusted Properties	0
		8	Clock reset control module trusted attribute 0	
		7	PMU Trusted Properties	0
		6	BT Trusted Properties	0
		5	GPSEC Trusted Properties	0
		4	DMA Trust Properties	0
		3	RSA Trusted Properties	0
		2	PSRAM Controller Trusted Properties	0
		1	FLASH Controller Trusted Properties	0

		0	SDIO HOST Trusted Properties	0
--	--	---	------------------------------	---

28.4 Instructions for use

28.4.1 Safe Memory Access (SASC)

28.4.1.1 ACCESS TO REGISTERS

Access to the SASC register follows the following principles:

- ÿ The CAR register can only be read and written when the cpu is security-super.
- ÿ When the cpu is in un-security-super, you can access the region-related registers configured as un-security-user

device position.

- ÿ When the cpu is security-super, all registers can be read and written.

- ÿ The registers are otherwise inaccessible

For example, when CAR is set to 16'he4e4, it means that region0 and region4 are set to un-security-user. At this time, if the cpu

In un-security-super, the cpu can read and write registers REGION0 and REGION4, as well as CR[0], CR[4], APx[1:0], APx[9:8], CDx[1:0], CDx[9:8].

28.4.1.2 Setting of protection area address

Each SASC supports 8 configurable memory protection range regions, and the base address in REGIONx[31:8] is the one you want to configure

25 to 2 bits of the actual physical address of the storage area. At the same time, the size and base address of Size need to meet the following requirements:

SIZE

00101: 4B;

00110: 8B; Baddrx[0] = 0

00111: 16B; Baddrx[1:0] = 0

01000: 32B; Baddrx[2:0] = 0

01001: 64B; Baddrx[3:0] = 0

01010: 128B; Baddrx[4:0] = 0

01011: 256B; Baddrx[5:0] = 0

01100: 512B; Baddrx[6:0] = 0

01101: 1KB; Baddrx[7:0] = 0

01110: 2KB; Baddrx[8:0] = 0

01111: 4KB; Baddrx[9:0] = 0

10000: 8KB; Baddrx[10:0] = 0

10001: 16KB; Baddrx[11:0] = 0

For example, if 20000100 is used as the base address of region0, then the maximum size of region0 can be set to 256B.

region0 is set to 128B, and the REGION0 register should be filled with 0x0000400a. If 20040000 is used as the base address, the region

The maximum can be set to 64KB. If a 64KB region is to be defined, the register should be filled with 0x01000013.

28.4.1.3 Access rights to memory

Priority of the four authority for mem:

request mem	Security super	Security user	Un-Security super	Un-Security user
Security super	✓ +	✓ +	✓ +	✓ +
Security user	X	✓	X	X
Un-Security super	X	X	✓	✓ +
Un-Security user	X	X	X	✓

Note: ✓ + all access, include read, write, data access, opcode fetch

✓ access based on the current attribute

X no access

According to the picture above

ÿ The bus access of security-super can access sram at will

ÿ The bus access of un-security-super can be freely configured as un-security-user

region

ÿ The bus access of security-user can only access security-user according to the current AP and CD properties

region

ÿ The bus access of un-security-super can access un by current AP and CD attributes

region of security-super

ÿ The bus access of un-security-user can only access un according to the current AP and CD attributes

security-user region

APx and CDx registers are used to set the access attributes when each region corresponds to different permissions, where CDx registers

The register is only valid for read operations during bus access, that is, to set whether to allow read data and read commands. APx registers are used for

Set whether to allow read/write operations.

For example, if CAR[1:0] is set to 00, and CR[0] is 1, it means REGION 0 is un-security-user, and

And enable permission protection, at this time, if the bus access is security-super or un-security-super, you can follow the

intended access to REGION 0; if the bus access is Security-user, any access will be denied; if the bus access is

Ask is un-security-user, AP0[1:0] is 01, which means read-only, CD0[1:0] is 01, which means data access, then

REGION 0 allows the bus to read data, other operations are denied.

If the AHB bus accesses the inaccessible area or the permission is wrong, the sasc module will give read deny

or write deny signal, thus returning an incorrect HRESP response signal, if the CPU is the one that initiated the bus access, then

A hardware exception will be generated, and if it is other devices, access will only be denied.

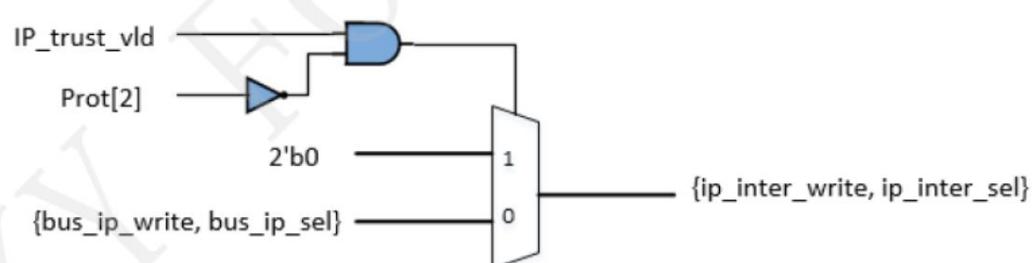
28.4.2 Trusted Access to Peripherals

The TIPC module can only be written when the PROT[2] signal is 1, that is, only in the trusted world can the IP address of each IP be written.

The letter attribute configuration register. The lp_trust_vld register is 0 by default, that is, all peripherals are untrusted. At this time, the peripherals

It can be accessed at will. If the ip_trust_vld corresponding to the peripheral is set, that is, the peripheral is set as a trusted device, only

Commands from the trusted world can access this peripheral.



29 Appendix 1. Chip Pin Definition

29.1 Chip Pinout

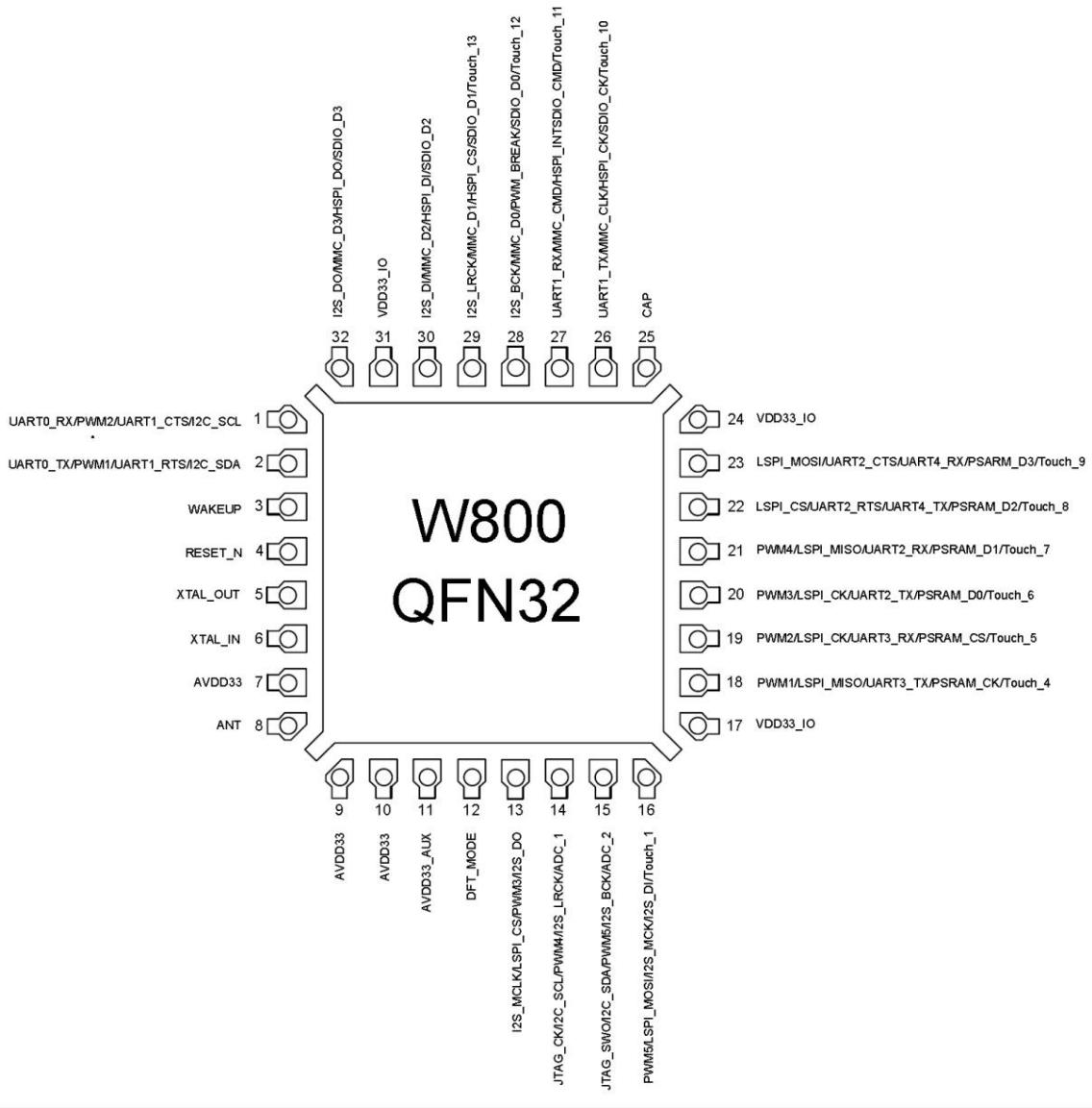


Figure 38 W800 chip pinout

Winner Micro

29.2 Chip pin multiplexing relationship

Table 220 Chip pin multiplexing relationship

Number	name	type	Pin function after reset	Multiplexing function	Maximum frequency	pull-up and pull-down	capability drive capability
1	PB_20	I/O UART_RX		UART0_RX/PWM2/UART1_CTS/I2C_SCL	10MHz	UP/DOWN	12mA
2	PB_19	I/O UART_TX		UART0_TX/PWM1/UART1_RTS/I2C_SDA	10MHz	UP/DOWN	12mA
3	WAKEUP	I	WAKEUP wake-up function			DOWN	
4	RESET	I	RESET reset			UP	
5	XTAL_OUT	O	External crystal output				
6	XTAL_IN	I	External crystal input				
7	AVDD33	P	chip power supply, 3.3V				
8	ANT	I/O	RF Antenna				
9	AVDD33	P	chip power supply, 3.3V				
10	AVDD33	P	chip power supply, 3.3V				
11	AVDD33_AUX	P	chip power supply, 3.3V				
12	TEST	I	Test function configuration pins				
13	BOOTMODE	I/O	BOOTMODE	I2S_MCLK/LSPI_CS/PWM3/I2S_DO	20MHz	UP/DOWN	12mA
14	PA_1	I/O	JTAG_CK	JTAG_CK/I2C_SCL/PWM4/I2S_LRCK/ADC_1	20MHz	UP/DOWN	12mA
15	PA_4	I/O	JTAG_SWO	JTAG_SWO/I2C_SDA/PWM5/I2S_BCK/ADC_2	20MHz	UP/DOWN	12mA
16	PA_7	I/O	GPIO, input, high impedance	PWM5/LSPI_MOSI/I2S_MCK/I2S_DI/Touch_1	20MHz	UP/DOWN	12mA

17	VDD33IO	P	IO power supply, 3.3V				
18	PB_0	I/O GPIO	input, high impedance	PWM1/LSPI_MISO/UART3_TX/PSRAM_CK/Touch_4 80MHz UP/DOWN		12mA	
19	PB_1	I/O GPIO	input, high impedance	PWM2/LSPI_CK/UART3_RX/PSRAM_CS/Touch_5	80MHz UP/DOWN	12mA	
20	PB_2	I/O GPIO	input, high impedance	PWM3/LSPI_CK/UART2_TX/PSRAM_D0/Touch_6	80MHz UP/DOWN	12mA	
twenty one	PB_3	I/O GPIO	input, high impedance	PWM4/LSPI_MISO/UART2_RX/PSRAM_D1/Touch_7 80MHz UP/DOWN		12mA	
twenty two	PB_4	I/O GPIO	input, high impedance	LSPI_CS/UART2_RTS/UART4_TX/PSRAM_D2/Touch_8	80MHz	UP/DOWN	12mA
twenty three	PB_5	I/O GPIO	input, high impedance	LSPI_MOSI/UART2_CTS/UART4_RX/PSRAM_D3/Tou ch_9	80MHz	UP/DOWN	12mA
twenty four	VDD33IO	P	IO power supply, 3.3V				
25	CAP	I	External capacitor, 1μF		-		
26	PB_6	I/O GPIO	input, high impedance	UART1_TX/MMC_CLK/HSPI_CK/SDIO_CK/Touch_10 50MHz UP/DOWN		12mA	
27	PB_7	I/O GPIO	input, high impedance	UART1_RX/MMC_CMD/HSPI_INT/SDIO_CMD/Touch_11	50MHz	UP/DOWN	12mA
28	PB_8	I/O GPIO	input, high impedance	2S_BCK/MMC_D0/PWM_BREAK/SDIO_D0/Touch_12 50MHz UP/DOWN		12mA	
29	PB_9	I/O GPIO	input, high impedance	I2S_LRCK/MMC_D1/HSPI_CS/SDIO_D1/Touch_13	50MHz UP/DOWN	12mA	
30	PB_10	I/O GPIO	input, high impedance	I2S_DI/MMC_D2/HSPI_DI/SDIO_D2	50MHz UP/DOWN	12mA	
31	VDD33IO	P	IO power supply, 3.3V				
32	PB_11	I/O GPIO	input, high impedance	2S_DO/MMC_D3/HSPI_DO/SDIO_D3	50MHz UP/DOWN	12mA	
33	GND	P ground					

statement

Beijing Lianshengde Microelectronics Co., Ltd. reserves the right to modify and update Lianshengde Microelectronics products or various documents and materials at any time. Place

Updates will be released through the official channels of Lianshengde Microelectronics. Users must ensure that the correct information is obtained through official channels. United Sidley

Microelectronics does not undertake to notify everyone of updated information.