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

				4’b0000ÿ Transaction of function 0 (CIA) 4’b0001-4’b0111ÿ Transaction to functions 1-7 4’b1000ÿ Transaction of memory in combo card 4’b1001-4’b1111ÿ 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] ÿ RFU	
0X14- 0XEF	RFU		RO	Reserved for Future Use (RFU)	8'b0
0XF0- 0XFF	Reserved for Vendors		RO	Area Reserved for Vendor Unique Registers	8'b0
0X100	I/O Device Interface Code		RO	[3:0], what kind of device is the flag Fn1. 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

			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 ÿ 2] ÿ 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 starting address. The storage method is LSB segment format.	17'h02 000
		RO	[23ÿ17], RFU	7'b0
0X10C · 0X10E	Address pointer to function CSA	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 Write 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 0- 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.		

0X2133				
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 address width of the AHB bus inside the chip is 32 bits, SDIO cannot directly access the inside of the chip using the 17-bit address.

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		[7:1]	RO	RSV	7'b0
	int_read_data	[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		[7:1]	RO	RSV	7'b0
	int_mask	[0]	RW	corresponds to the mask enable signal of int_src. 1 to mask the corresponding interrupt.	1'd0
0X06		[7:2]	RO	RSV	6'b0
	wlan_awake_stts [0]		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	RSV	3'b0
0X1E			RO	RSV	
0X1F		[7:2]	RO	RSV	6'b0
	down_cmdbuf_vld	[1]	RO	downlink command buffer is available, 1 is valid.	1'b1
	txbuf_vld	[0]	RO	downstream data buffer is available. 1 is valid, indicating that there is an available send bufferÿ	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 transmitting data, it needs to be used in conjunction with the wrapper controller. For the part of the wrapper controller, please refer to the HSPI part of the documentation.

Table 93 SDIO AHB bus registers

offset address name		Bit wide access description			reset value
0X0000 0X0004			RO	Rsv	
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 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

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

				1: Support [26], SHS, default is 1, support high speed 0: not supported 1: Support [31:27]: RFU	
0X0028	Program Register		RW	[0], function ready. hc8051 is required to complete SD initialization After the desired Fn1 register is assigned, set this register to send the SD host to the Indicates that Fn1 is ready to work. Default is 0 0:Fn1 not ready 1ÿFn1 ready [1], fun1 read data ready. Fn1 is ready to send to SD host When sending data, set this register. After the host responds to the read interrupt, the read Interrupt source register (Interrupt Identification), this bit from Move to zero. Default is 0. 0: No data is sent to the host 1: There is data sent to the host. [2], SCSI. Continuous SPI interrupts are supported. The default is 1. 0: not supported 1: Support [3], SDC. Indicates that the SDIO card supports execution at the same time as data transmission CMD52 command. Default is 1 0: not supported	16'h 02fc

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

				[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'h00ff80 00

				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 21 3.3-3.4 22 3.4-3.5 23 3.5-3.6 [31:24]ÿ8'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

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

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 the specific length. 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

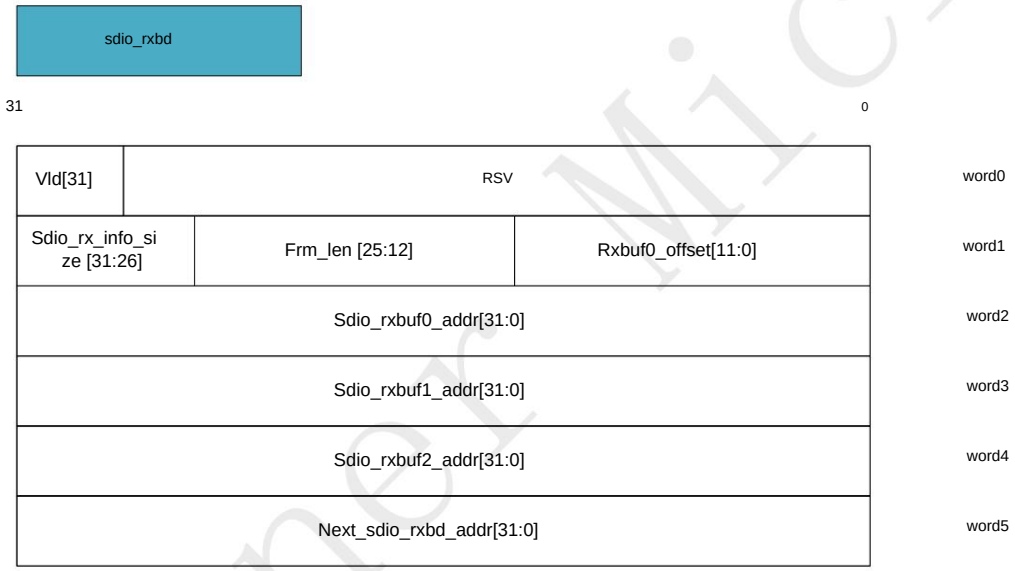
12.3.1 Uplink data receiving function

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:



Description: 1.vld, 1 is valid, indicating that the current descriptor points to a valid received frame. 2.rxbuf0_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. Rxbuf0_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.

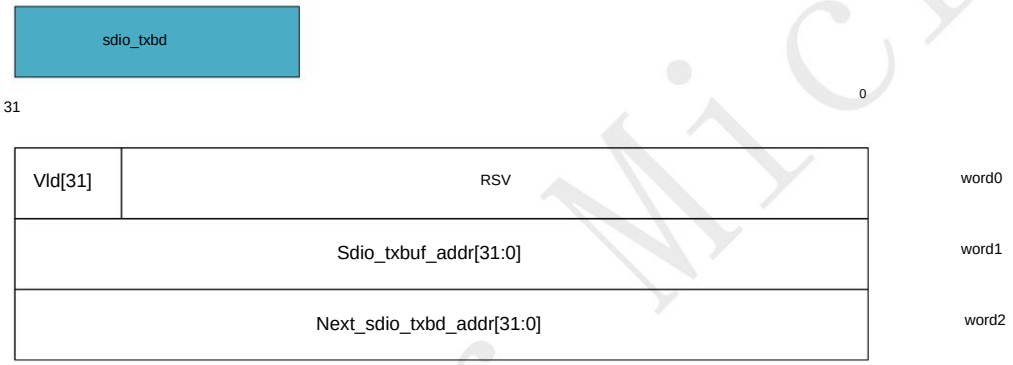
12.3.2 Downlink data transfer function

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

Rank	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

Rank	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

Rank		access	Instructions	reset value

[31: 1]		RO	Reserve	
[0]		RW	Firmware sets this bit to 1 when there is an upstream cmd frame. When the upstream cmd transfer is complete, the hard It is automatically cleared to 0 by the event and an int_up_cmd interrupt is generated.	1'b0

12.4.5 WRAPPER downlink command buf ready register

Table 98 WRAPPER downlink command buf ready register

Rank	access	Instructions	reset value
[31: 1] RO		Reserve	
[0]	RW	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 After command, set this bit to 1.	1'b0

12.4.6 SDIO TX Link Enable Register

Table 99 SDIO TX Link Enable Register

Rank	access	Instructions	reset value
[31: 1] RO		Reserve	
[0]	RW	sdio_txbd link enable, active high. 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 After sdio_txbd, it will stop immediately. The same applies to HSPI.	1'b1

12.4.7 SDIO TX Link Address Register

Table 100 SDIO TX Link Address Register

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

12.4.8 SDIO TX Enable Register

Table 101 SDIO TX enable register

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

12.4.9 SDIO TX Status Register

Table 102 SDIO TX Status Register

Rank	access	Instructions	reset value

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

12.4.10 SDIO RX Link Enable Register

Table 103 SDIO RX Link Enable Register

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

12.4.11 SDIO RX Link Address Register

Table 104 SDIO RX Link Address Register

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

		The same applies to HSPI.	

12.4.12 SDIO RX Enable Register

Table 105 SDIO RX enable register

Rank	access	Instructions	reset value
[31: 1] RO		Reserve	
[0]	RW	SDIO receive frame enable, active high. The firmware sets this bit to 1 after completing each descriptor sdio_rxbd descriptor to notify the SDIO module 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. Hardware automatically completes the clearing of this register. The same applies to HSPI.	1'b0

12.4.13 SDIO RX Status Register

Table 106 SDIO RX Status Register

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

12.4.14 WRAPPER CMD BUF BASE ADDRESS REGISTER

Table 107 WRAPPER CMD BUF Base Address Register

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

12.4.15 WRAPPER CMD BUF SIZE register

Table 108 WRAPPER CMD BUF SIZE register

Rank	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 Abortion	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]	Set automatic 8 null clock/command/response transmission Features '1': Enable automatic 8 null clock/command/response transmission '0': Disable automatic 8 null clock/command/response transmission lose. Generate 8 nulls according to bit 5 and bit3 settings Clock, receive response or transmit command. when the biography After the input is completed, this bit is automatically cleared;	1'b0
			RW	[1]	Set data transfer direction '1' : read data; . '0' : write data;	1'b0
			RW	[0]	Set up automatic data transfer '1' : enable automatic data transfer '0' : Disable automatic data transfer. When the data transmission is completed, this bit will be automatically cleared;	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]	SD/MMC/SDIO port CMD Line CRC circuit enable. SD/MMC/SDIO port CMD line CRC function	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] 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]	W: Clear command and respond to Ncr timeout interrupt; R: Command and response Ncr timeout interrupt status;	1'b0
			RW	[5]	W: clear multi-block timeout interrupt; . R: Multi-data block timeout interrupt status;	1'b0
			RW	[4]	W: Clear multi-block transfer completion interrupt; . R: Multi-data block transfer completion interrupt status;	1'b0
			RW	[3]	W: Clear command CRC error interrupt; . R: Command CRC error interrupt status;	1'b0
			RW	[2]	W: clear data CRC error interrupt; . R: Data CRC error interrupt status;	1'b0
			RW	[1]	W: Clear data transfer completion interrupt; . R: Data transmission complete interrupt status;	1'b0
			RW	[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; 1MHz=Fhclk/((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]	Set SD/MMC/SDIO port auto multiple block data ransfer. Set SD/MMC/SDIO port automatic multi-blocking transmission '1' : enable. '0' : off. Setting this bit to 1 (mmc_io[7:6]==11) will trigger Send a multi-block transfer of SD/MMC/SDIO. The number of data blocks is in mmc_blocknt set in the register. When the data transmission is completed, this bit will Automatically clear.	1'b0
0x38	mmc_blockcnt		RW	[15:0]	Data block number register. data block number register Configuring this register defines a total of The number of data blocks to be transmitted.	16'h0001
0x3C	mmc_timeoutcnt		RW	[7:0]	Data transfer timeout count register. Data transfer timeout counter Time = Scale* bit[7:0]. Scale by register mmc_io_mbctl[7:6]/[5:4] definition;	8'h40
0x40	cmd_buf0		RW	[7:0]	The cmd_buf byte 0. Mapped to command line bit [15:8]	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)ÿ

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_bytecnt
x	x	x	x	x	0	0	Trig write	single data +8 null clk	mmc_bytecnt

Table 3 Mmc_io[7:0] detailed definition

Noteÿ

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

Noteÿ

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

Noteÿ

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] ÿ

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

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

String. 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 is used to perform some configure		0X0000_0000
0X0004	SPI configuration register SPI_CFG	RW	Configure SPI communication related items	0X0000_0004
0X0008 Clock configuration register Clk_CFG		RW is used to set the clock frequency division factor		0X0000_0000
0X000C Mode configuration register MODE_CFG		RW configuration transfer mode		0X0000_0000
0X0010 Interrupt Control Register SPI_INT_MASK		RW mask or enable related interrupt		0X0000_00FF
0X0014 Interrupt status register SPI_INT_SOURCE RW is used to query the interrupt source				0X0000_0000
0X0018	SPI Status Register SPI_STATUS	RO enumerates relevant states in SPI communication		0X0000_0000
0X001C	SPI Timeout Register SPI_TIME_OUT	RW Set SPI communication timeout		0X0000_0000
0X0020 Data transmission register SPI_TX_DATA		RW	TX FIFO, used to store the data to be sent 0X0000_0000	
0X0024 Transfer mode register TRANS_MODE		RW Set transfer mode		0X0000_0000
0X0028 Data length register SLV_XMIT_LEN		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 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

		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 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 Note: master mode is valid Motorola/TI mode active	
[22]	RW	Clear FIFOs, clear the contents of Tx and Rx FIFO, and simultaneously reset all circuits of master and slave (except configuration registers) 1'b0: do not clear the FIFO 1'b1: Clear valid Set to 1 by software, cleared to 0 by hardware Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1'b0
[21]	RW	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. 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 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 Note: master is valid Normally, this mode is not set.	1'b0

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

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

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

		1'b0: slave, the device is slave 1'b1: master, the device is the master Note: Both master/slave are valid	
[1]	RW	SPI CPHA 1'b0: Transmission Mode A 1'b1: transfer mode B Note: Both master/slave are valid Motorola mode is valid	1'b0
[0]	RW	SPI CPOL, SCK polarity at IDLE 1'b0: 0 when SCK IDLE 1'b1: 1 for SCK IDLE Note: Both master/slave are valid Motorola mode is valid	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		Divider FSCK = FAPB_CLK/ (2 x (Divider +1)) Note: master is valid Motorola/TI/microwire mode is valid	16'h0
[31:16]		RSV	16'h0

[15:0] RW		Divider FSCK = FAPB_CLK/ (2 x (Divider +1)) Note: master is valid Motorola/TI/microwire mode is valid	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		RxTrigger level Data stored in RX FIFO triggers interrupt or DMA request threshold: 0~7word Only when the data in the rxbuffer is greater than the RxTrigger level will trigger an interrupt or request a DMA transfer Note: Both master/slave are valid Motorola/TI/microwire mode is valid	3'b0
[5]		RSV	1'b0
[4:2] RW		TxTrigger level Data stored in TX FIFO triggers interrupt or DMA request threshold: 0~7word 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. shift Note: Both master/slave are valid Motorola/TI/microwire mode is valid	3'b0
[1]	RW	RxDMA On, use DMA to move data enable	1'b0

		1'b0: Do not use DMA, 1'b1: DMA Note: Both master/slave are valid Motorola/TI/microwire mode is valid	
[0]	RW	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	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	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	1'b1
[6]	RW	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	1'b1

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

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

14.4.7 Interrupt Status Register

Table 115 SPI Interrupt Status Register

bit access		Instructions	reset value
[31:8]		RSV	24'h0
[7]	RW	spi_timeout 1'b0: There is no end data in rxfifo that needs to be taken by the CPU 1'b1: There is end data in rxfifo that needs to be taken by the CPU Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1'b0
[6]	RW	spi_done 1'b0: SPI transmission or reception is not completed 1'b1: SPI send or receive completed Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1'b0
[5]	RW	RxOverrun 1'b0ýRx FIFO overflow 1'b1ýRx FIFO overflow Write 1 to clear	1'b0

		Note: Both master/slave are valid Motorola/TI/microwire mode is valid	
[4]	RW	RxUnderrun 1’b0ÿRx FIFO underflow 1’b1ÿRx FIFO underflow Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1’b0
[3]	RW	TxOverrun 1’b0ÿTx FIFO overflow 1’b1ÿTx FIFO overflow Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1’b0
[2]	RW	TxUnderrun 1’b0ÿTx FIFO underflow 1’b1ÿTx FIFO underflow Write 1 to clear With continue mode = 1, the interrupt is never generated. Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1’b0
[1]	RW	RxFifoRdy 1'b0: Rx FIFO data volume <= RxTrigger level, no need to upload 1'b1: Rx FIFO data volume > RxTrigger level, request to upload Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1’b0

[0]	RW	TxFifoRdy 1'b0: Tx FIFO data volume > TxTrigger level, cannot write data to TX FIFO 1'b1: Tx FIFO data volume <= TxTrigger level, data can be written to TX FIFO Write 1 to clear Note: Both master/slave are valid Motorola/TI/microwire mode is valid	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	SPI Busy 1'b0: SPI has no transmit and receive tasks 1'b1: SPI is in the process of sending or receiving Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1'b0
[11:6] RO		RX FIFO fill level The amount of data in the Rx FIFO, in bytes Note: Both master/slave are valid Motorola/TI/microwire mode is valid	6'h0
[5:0]	RO	Tx FIFO fill level The amount of data in the Tx FIFO, in bytes Note: Both master/slave are valid Motorola/TI/microwire mode is valid	6'h0

14.4.9 SPI Timeout Register

Table 117 SPI Timeout Register

bit access		Instructions	reset value
[31]	RW	spi_timer_en 1'b0: Do not allow timer timing 1'b1: allow timer timing Note: Both master/slave are valid Motorola/TI/microwire mode is valid	1'b0
[30:0] RW		SPI_TIME_OUT When a transmission is completed, in the receiving channel rxfifo, if the data at the end cannot trigger the receiving interrupt When RxFifoRdy or DMA request, a timing mechanism needs to be used to notify the CPU to remove the end data. 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. If the value is set, the timeout interrupt is triggered, and the CPU is requested to remove the end data. Any read or write to rxfifo will clear the timeout timer. The time represented is: T = SPI_TIME_OUT/FAPB_CLK Note: Both master/slave are valid Motorola/TI/microwire mode is valid	31'h0

14.4.10 Data Transmission Register

Table 118 SPI data transmission register

bit access		Instructions	reset value

[31:0] RW		Window address for writing data to Tx FIFO Note: Both master/slave are valid Motorola/TI/microwire mode is valid	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		TI_BLK_LEN In the timing mode of TI, the length of each block transmission, that is, the transmission data length after each CS is valid. Support 4~32bit 6'h4: 4bit long data 6'h5: 5bit long data 6'h6: 6bit long data …….. 6'h20: 32bit long data Note: master is valid TI mode active	6'd0
[16]	RW	MICRO_BURST 1b'1: In Microwire mode, burst transmission is used, that is, Tx sends the control word, Rx receives the data, and so on. Alternately, MICRO_CONTROL_LEN indicates the length of the control word, MICRO_DAT_LEN indicates	1'b0

		is the length of the sent or received word, Tx/Rx length indicates the effective sck, burst during the entire transmission process In mode, the number of times of sending and receiving alternately is (Tx/Rx length)/(MICRO_CONTROL_LEN+ MICRO_DAT_LEN+1) 1'b0: In Microwire mode, burst transmission is not used In this mode, there are two cases 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 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	
[13:8] RW		MICRO_DAT_LEN In Microwire mode, in burst mode, the length of each burst transmission data From 1 to 32: 6'h1: 1bit long data 6'h2: 2bit long data	6'd0

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

14.4.12 Data Length Register

Table 120 SPI Data Length Register

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

under these circumstances. 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	PRER it	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		The clock divider configures the lower 8bits of prescale. For example: apb_clk=40MHz, SCL=100KHz prescale = (40*1000)/(5*100) – 1 = 16'd79 apb_clk = 40M, SCL=400K 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		The clock divider configures the high 8bits of prescale. For example:	8'hff

		apb_clk=40MHz, SCL=100KHz prescale = (40*1000)/(5*100) – 1 = 16'd79 apb_clk = 40M, SCL=400K prescale=(40*1000)/(5*400) – 1 = 16'd19	

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

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

15.4.6 Transceiver Control Register

Table 127 I2C Transceiver Control Register

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

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

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

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 through

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.

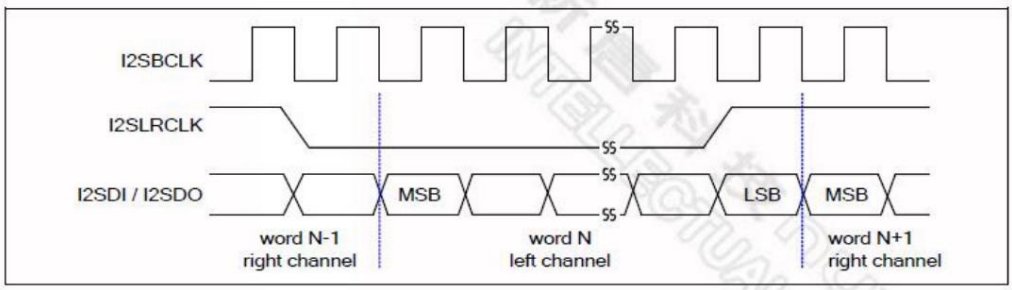


Fig1. I2S Bus Timing DiagramÿPCM=0ÿFormat=0ÿ

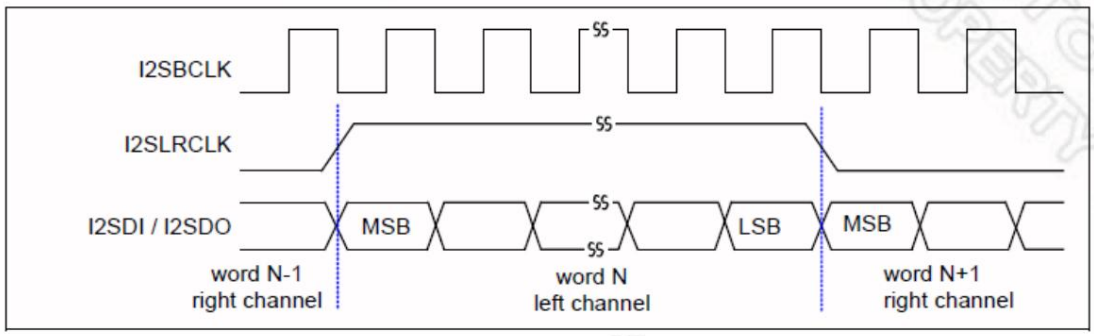


Fig2. MSB Justified Timing DiagramÿPCM=0ÿFormat=1ÿ

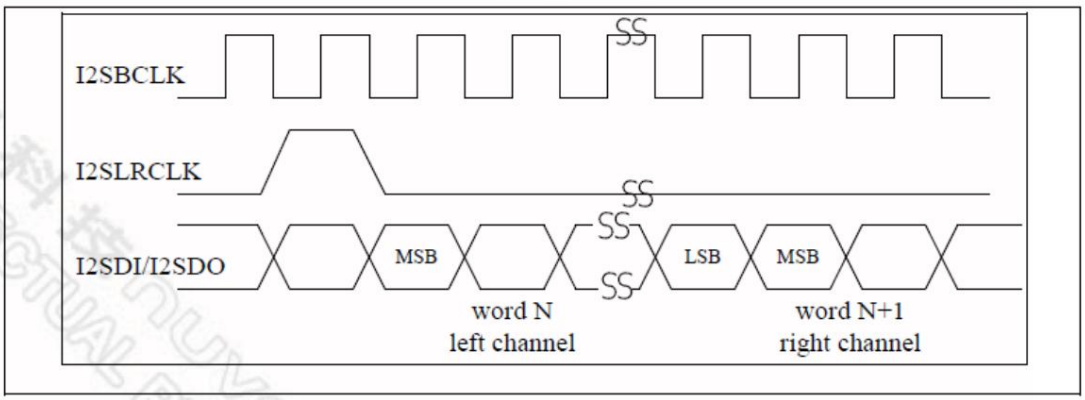


Fig3. PCM A Audio DiagramÿPCM=1ÿFormat=0ÿ

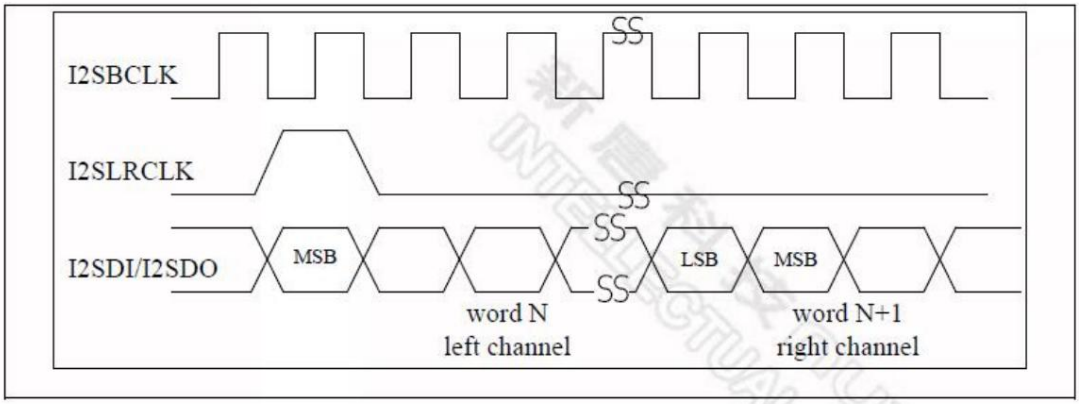


Fig4. PCM B Audio DiagramÿPCM=1ÿFormat=1ÿ

16.5 FIFO storage structure diagram

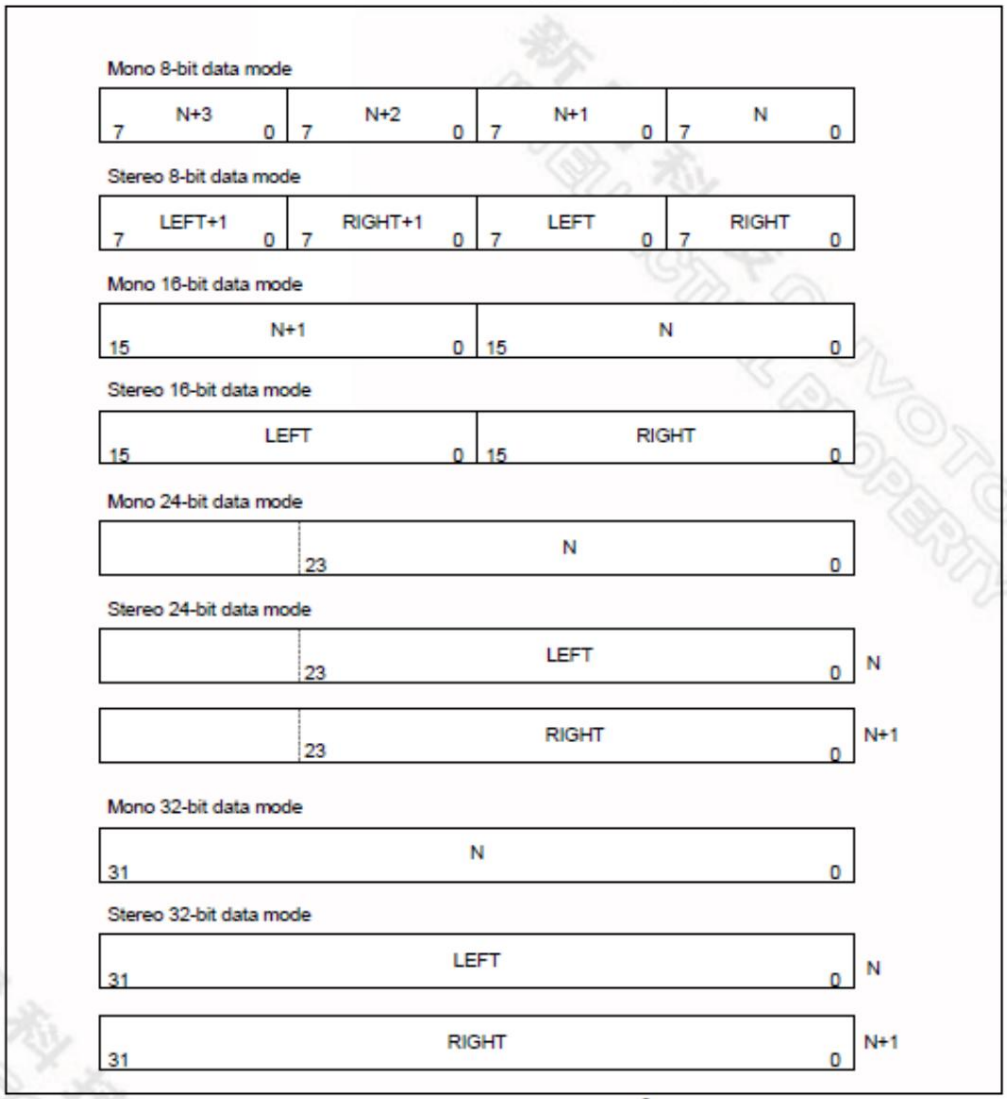


Fig 5. FIFO storage structure

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>=2;

Bits[17:8] of register 0x40000718 is the area for configuring the divider ratio of the BLCK 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=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'd26ÿ

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*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 >=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	BCLKDIV BCLK divider F_BCLK=F_I2SCLK/BCLKDIV Note: When EXTAL_EN is not selected, Internal PLL is used and F_I2SCLK =160MHz	10'd0

				When WXTAL_EN is enabled, F_I2SCLK = External crystal frequency **BCLKDIV=round (F_I2SCLK/(Fs*W*F))** Where Fs is sample frequency of audio data and W is word width. F=2 when data is stereo and F=1 when data is mono. For example, if internal PLL is used and the data width is 24bit, Format is stereo format, sample frequency is 128KHz. Then the BCLKDIV should be configured as (160*10e6/128*10e3*24*2) =10'd26	
		[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 interrupt 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
[23]	RW	RXLCH	1'b0

		Channel receive enable control bit 1'b0: Enable to receive right channel data 1'b1: Enable to receive left channel data Note: This bit is only valid when the mono mode of MONO_STEREO is selected	
[22]	RW	MONO_STEREO Mono Stereo Select Bits 1'b0: Data is transmitted in stereo format 1'b1: data is transmitted in mono format	1'b0
[21]	RW	RXDMAEN Receive DMA request enable bit 1'b0: Disable send DMA request 1'b1: Enable send DMA request 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.	1'b0
[20]	RW	TXDMAEN Send DMA request enable bit 1'b0: Disable send DMA request 1'b1: Enable send DMA request Note: When the transmit DMA request is enabled and the number of words in the TXFIFO is less than TXTH, the I2S controller A transfer request will be issued to the DMA until the TXFIFO is full and the DMA transfer will not be stopped.	1'b0
[19]	WHERE	RXCLR clear RXFIFO	1'b0

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

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

[7:6] RW		RSV	2'h0
[5:4] RW		WDWIDTH Transfer word length setting bits 2'b00: word length 8 bits 2'b01: word length 16 bits 2'b10: word length 24 bits 2'b11: word length 32 bits	2'b0
[3]	RW	MUTE Transmit mute enable flag 1'b0: transfer data from shift register, normal operation mode 1'b1: Set transmit data to 0, mute the sound	1'b0
[2]	RW	RXEN receive enable flag 1'b0: Stop I2S data reception 1'b1: Enable I2S data reception	1'b0
[1]	RW	TXEN transfer enable flag 1'b0: Stop I2S data transmission 1'b1: Enable I2S data transfer	1'b0
[0]	RW	I2SEN I2S enable flag 1'b0: Disable	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	LZCIMASK Left channel zero crossing interrupt enable flag 1'b0: Interrupt disabled 1'b1: Interrupt enable An interrupt is generated when the interrupt is enabled and a zero crossing is detected on the left channel	1'b1
[8]	RW	RZCIMASK Right channel zero crossing interrupt enable flag 1'b0: Interrupt disabled 1'b1: Interrupt enable An interrupt is generated when the interrupt is enabled and a zero crossing is detected on the right channel	1'b1
[7]	RW	TXDONEMASK Transmit complete interrupt enable bit 1'b0: Interrupt disabled 1'b1: Interrupt enable An interrupt is generated when the interrupt is enabled and the TXFIFO is empty	1'b1
[6]	RW	TXTHIMASK	1'b1

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

		1'b0: Interrupt disabled 1'b1: Interrupt enable Generated when the RXFIFO threshold interrupt is enabled and the number of data in the RXFIFO is equal to or more than the threshold Generate RX interrupt	
[1]	RW	RXOVIMASK RXFIFO overflow interrupt enable bit 1'b0: Interrupt disabled 1'b1: Interrupt enable Note: When the RXFIFO outflow interrupt is enabled and TXOVIF is detected as 1, an overflow interrupt will be generated	1'b1
[0]	RW	RXUDIMASK RXFIFO underflow interrupt enable bit 1'b0: Interrupt disabled 1'b1: Interrupt enable Note: When the RXFIFO underflow interrupt is enabled and TXUDIF is detected as 1, an underflow interrupt will be generated	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	TXIF I2S transmit interrupt flag 1'b0: No I2S interrupt occurred	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	TXDONEIF Transmit complete interrupt flag 1'b0: This send is not completed 1'b1: This transmission is completed Note: write 1 to clear the interrupt flag	1'b0
[6]	RO	TXTHIF RXFIFO interrupt flag 1'b0: The number of words in the TXFIFO is greater than the threshold 1'b1: The number of words in the TXFIFO is less than or equal to the threshold. Note: When the number of words in the TXFIFO (TXCNT) is equal to or less than the threshold set by TXTH, this bit will Set to 1 until data is written to the TXFIFO and the value of TXCNT is greater than the value of TXTH change back to 0.	1'b0
[5]	RW	TXOVIF TXFIFO overflow interrupt flag 1'b0: TXFIFO overflow interrupt does not occur 1'b1: TXFIFO overflow interrupt occurred Note: write 1 to clear the interrupt flag	1'b0
[4]	RW	TXUDIF TXFIFO underflow interrupt flag 1'b0: No underflow interrupt occurred in TXFIFO 1'b1: TXFIFO underflow interrupt occurred Note: write 1 to clear the interrupt flag	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	VALIDBYTE The number of bytes available in the last word. 2'b00: After receiving, all bytes in RXFIFO are available 2'b01: 1 byte is available in RXFIFO after reception is complete 2'b10: 2 bytes are available in the RXFIFO after reception is complete 2'b11: 3 bytes are available in the RXFIFO after reception is complete	2'h0
[7:4]	RO	TXCNT Record the number of words in the TXFIFO at the current moment. 4'b0000: no data 4'b0001: has 1 word … 4'b1000: has 8 characters	4'h0
[3:0]	RO	RXCNT Record the number of words in the RXFIFO at the current moment. 4'b0000: no data 4'b0001: has 1 word … 4'b1000: has 8 characters	4'h0

16.9.6 Data Transmission Register

Table 135 I2S data transmission register

bit access		Instructions	reset value
[31:0] WO		TXFIFO 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.	32'h0

16.9.7 Data Receive Register

Table 136 I2S data receive register

bit access		Instructions	reset value
[31:0] RO		RXFIFO 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.	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 ÿ

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:

ubdiv = apbclk / (16 * baudrate) – 1 // Integer

ubdiv_frac = (apbclk % (baudrate * 16)) / baudrate //Integer

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

ubdiv = 40000000 / (16 * 19200) – 1 = 129

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:

BAUD_RATE _CTRL = (3<<16) | 129 = 0x0003_0081ÿ

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:

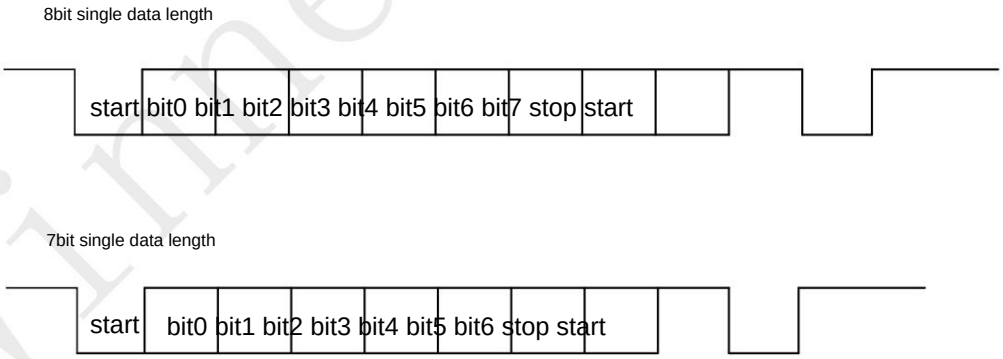


Figure 28 UART data length

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, which 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:

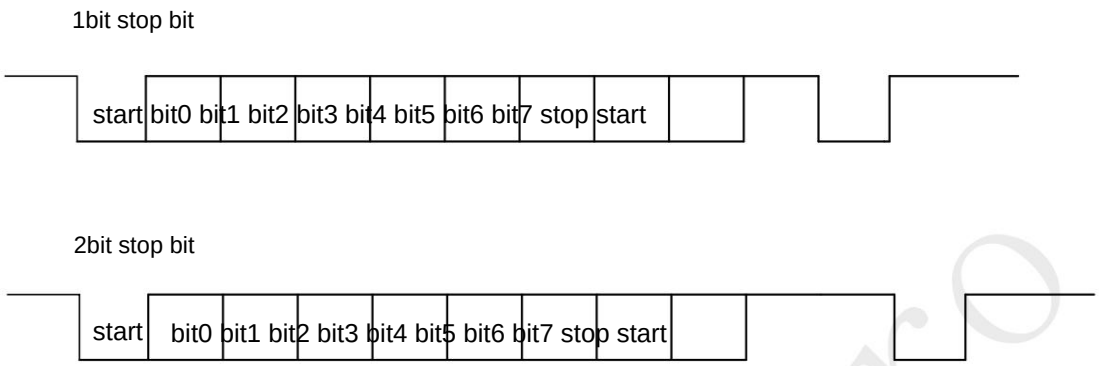


Figure 29 UART stop bit

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.

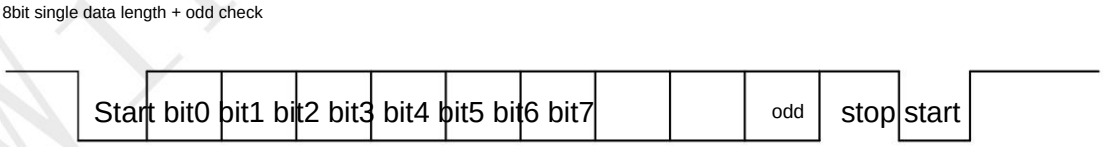
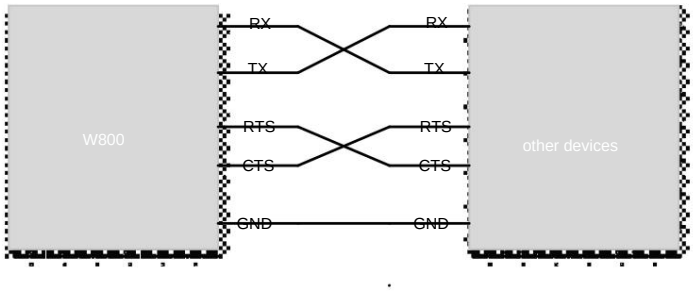


Figure 30 UART parity bit

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_WIDNOW, 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 It is automatically cleared to 0 after that.	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'h0ÿ5bit 2'h1ÿ6bit 2'h2ÿ7bit 2'h3ÿ8bit	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: 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	
[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		rxfifo timeout num When the data in rxfifo is less than rxfifo_trigger_level, if there is no data within N packets When new data is received, an rxfifo 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	rxfifo timeout en	1'b1

		rxfifo timeout enable	
[1]	RW	rx dma enable Receive DMA enable, active high. 0: Indicates that the receiving process uses interrupts.	1'b0
[0]	RW	tx dma enable Transmit DMA enable, active high. 0: Indicates that the sending process uses interrupts.	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 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. 2'h0ÿ1byte 2'h1ÿ4byte 2'h2ÿ8byte 2'h3ÿ16byte	2'h1
[3 : 2] RW		txfifo trigger level 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. 2'h0ÿempty 2'h1ÿ4byte	2'h1

		2'h2ÿ8byte 2'h3ÿ16byte	
[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 frac×16. (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. Software actively writes 1 to clear to 0.	1'b0
[7]	RW	parity error The received packet has an incorrect parity bit. In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt. Software actively writes 1 to clear to 0.	1'b0

[6]	RW	frame error Received packet with stop bit error. In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt. Software actively writes 1 to clear to 0.	1'b0
[5]	RW	break detect A break packet is received. In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt. Software actively writes 1 to clear to 0.	1'b0
[4]	RW	cts changed This interrupt is generated when the cts signal changes. Software actively writes 1 to clear to 0.	1'b0
[3]	RW	rxfifo data timeout The data length in rxfifo is less than rxfifo trigger level but no data is received for N data cycles, An interrupt is generated. Software actively writes 1 to clear to 0.	1'b0
[2]	RW	rxfifo trigger level interrupt When the number of data in rxfifo changes from less than the number specified in rxfifo trigger level to greater than or equal to the number , this interrupt is generated. At this point, the current data frame size should be determined according to the rxfifo count. Software actively writes 1 to clear to 0.	1'b0
[1]	RW	txfifo trigger level interrupt 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	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

17.4.10TX 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

		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.	

17.4.11RX Start Address Register

Table 147 UART RX Start Address Register

bit access		Instructions	reset value
[31: 0] RO		rx data window 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 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 Possible placement 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.

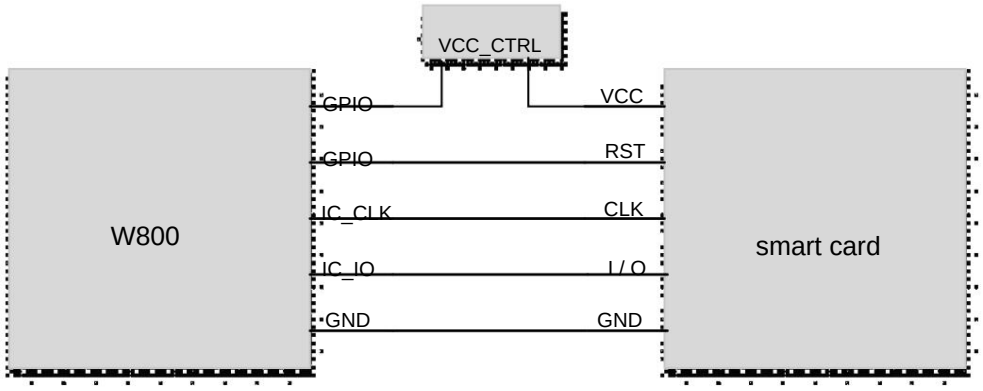


Figure 32 7816 Connection Diagram

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:

$$clk\_div = \frac{\_}{2 \times \_} \text{ ÿ } 1$$

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:

clk_div = (fclk_apb + fsc_clk)/(2 * 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 1etu = × — —

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 — *Fi* and *f* (max.)

Bits 8 to 5	0000	0001	0010	0011	0100	0101	0110	0111
Fi	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
Fi	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 *Di*.

Table 8 — *Di*

Bits 4 to 1	0000	0001	0010	0011	0100	0101	0110	0111
Di	RFU	1	2	4	8	16	32	64

Bits 4 to 1	1000	1001	1010	1011	1100	1101	1110	1111
Di	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

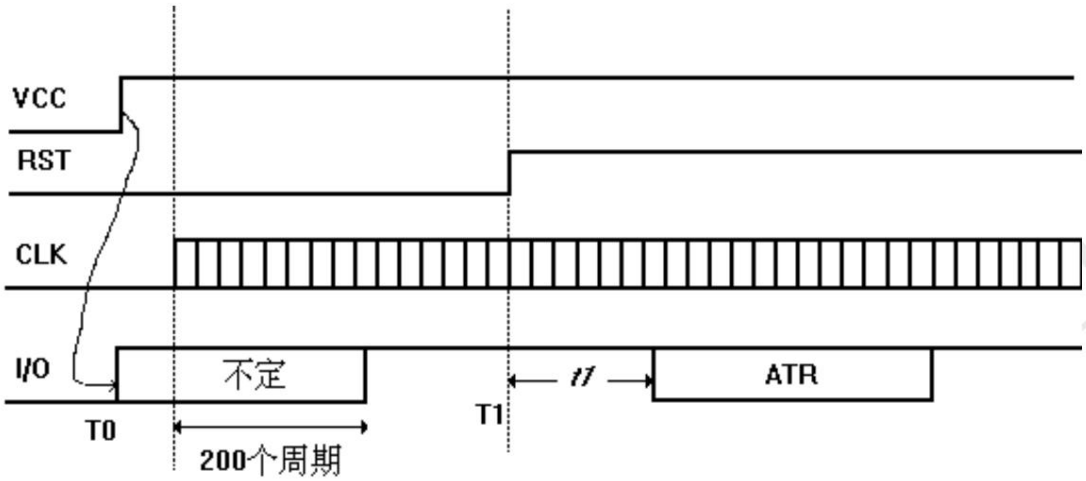


Figure 33 7816 power-on reset sequence

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

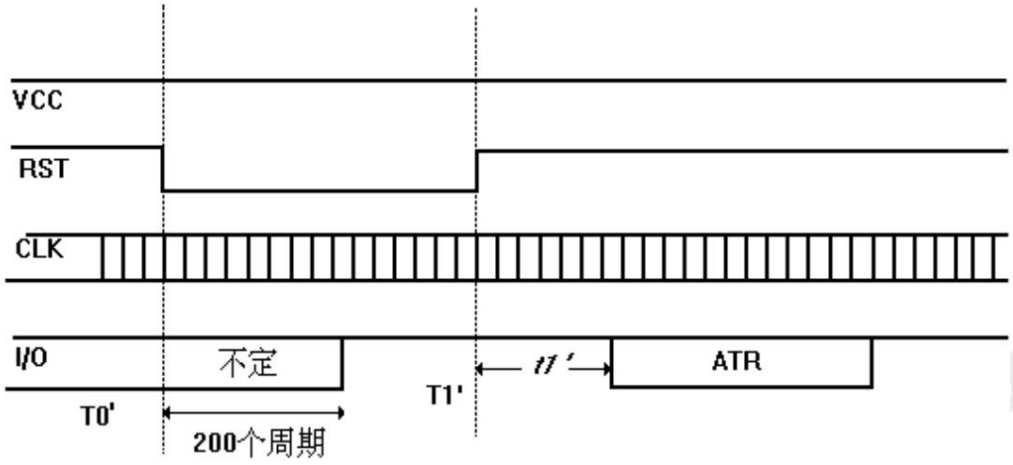


Figure 34 7816 warm reset

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

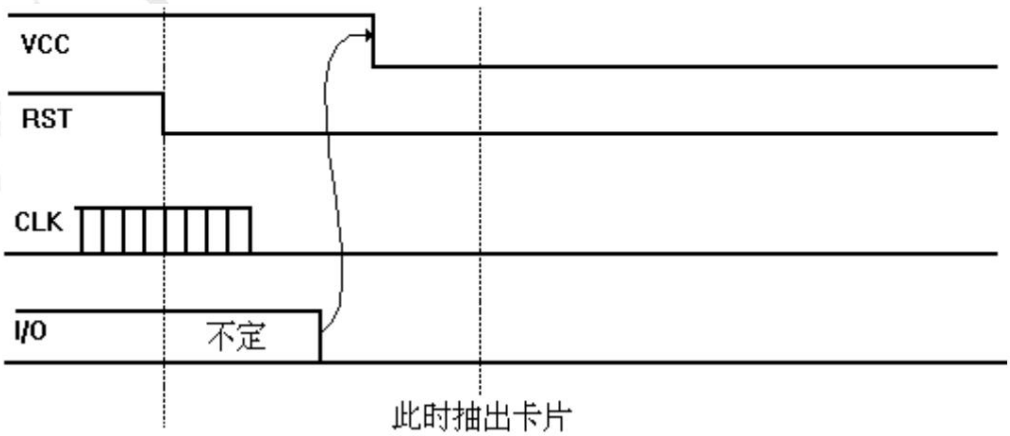


Figure 35 7816 inactivation process

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.

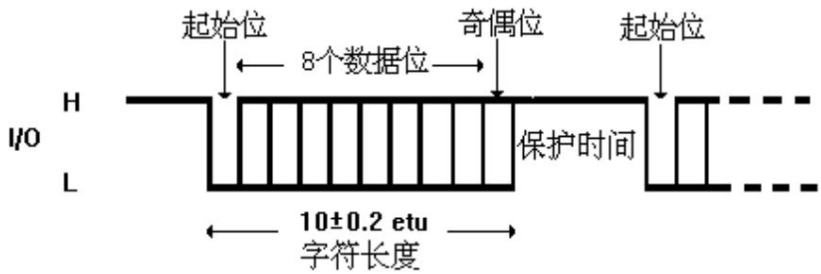


Figure 36 7816 data transmission

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

0ÿ

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	
[24]	RW	sc_mode 1'b0: uart mode 1'b1: 7816 mode	1'b0
[23]	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	7816 card T0 mode tx_retrans_en 1'b0: Tx automatic retransmission is invalid 1'b1: Tx automatic retransmission enable	1'b0
[18:16] RW		7816 card T0 mode tx_retrans_cnt tx automatic retransmission times	3'h3
[15:11] RW		The minimum MIN_BGT (Min Block Guard Time) of the 7816 card Min Block Guard Time calculation: 10+stop bits (default 2 bits)+configured value MIN_BGT Note: T=0: The minimum time interval between the falling edges of the start bits of consecutive characters in opposite directions of transmission and reception 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 The interval between the falling edges of the section start bit is 15 ETU characters. T=1: The minimum time interval between the falling edges of the start bits of consecutive characters in opposite directions of transmission and reception (Block Guard Time, BGT) must be 22 ETUs. must be able to interpret the received start bit correctly The interval between the falling edge and the falling edge of the last sent byte start bit is the character received within 21 ETUs.	5'ha
[10]	RW	7816 Card Clock Control Configuration 1'b0: The card clock output is generated when configured in card mode, otherwise the card clock output is invalid 1'b1: Clock stopped	1'b0
[9]	RW	Whether to receive data when the parity of the 7816 card is wrong 1'b0: do not receive 1'b1: receive	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 It is automatically cleared to 0 after that.	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'h0ÿ5bit 2'h1ÿ6bit 2'h2ÿ7bit 2'h3ÿ8bit	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		rxfifo timeout num (UART mode) When the data in rxfifo is less than rxfifo_trigger_level, if there is no data within N packets When new data is received, an rxfifo 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	rxfifo timeout en (UART&7816 mode) rxfifo 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		rxfifo trigger level (UART&7816 mode) 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. 2'h0ÿ1byte 2'h1ÿ4byte 2'h2ÿ8byte 2'h3ÿ16byte	2'h1
[3 : 2] RW		txfifo trigger level(UART&7816 ÿÿ) 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. 2'h0ÿempty 2'h1ÿ4byte 2'h2ÿ8byte 2'h3ÿ16byte	2'h1
[1]	RW	rxfifo reset (UART&7816 mode) Reset rxfifo, clear rxfifo state	1'b0
[0]	RW	txfifo reset (UART&7816 mode) Reset txfifo, clear txfifo state	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		ubdiv_frac UART mode: Indicates the fractional part of the system clock divided by 16 times the baud rate clock quotient. The specific value is frac×16. (Refer to chapter Baud rate calculation method) 7816 mode: ubdiv_frac = (fclk_apb + fsc_clk)/(2 * fsc_clk) - 1; (Refer to 7816 clock calculation method)	4'h3
[15: 0] RW		ubdiv UART mode: 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 the baud rate calculation method) 7816 mode: ubdiv=Fi/Di (Fi, Di are the parameters fed back by the smart card, edu frequency: f_etuclk = fsc_clk/(ubdiv+1)) (Refer to Section 7816 Rate Calculation Method)	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 model)	1'b1
[2]	RW	rxfifo trigger level int mask, rxfifo reaches the trigger value interrupt mask bit, active high. (UART&7816 mode Mode)	1'b1
[1]	RW	txfifo trigger level int mask, txfifo reaches the trigger value interrupt mask bit, active high. (UART&7816 mode 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	overrun error (UART&7816 mode) rxfifo overflowed. Software actively writes 1 to clear to 0.	1'b0
[7]	RW	parity error (UART&7816 mode) The received packet has an incorrect parity bit. In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt. Software actively writes 1 to clear to 0.	1'b0
[6]	RW	frame error (UART mode) Received packet with stop bit error. In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt. Software actively writes 1 to clear to 0.	1'b0
[5]	RW	break detect (UART mode) A break packet is received. In the case of DMA, this interrupt will still be generated. But DMA operations don't care about this interrupt. Software actively writes 1 to clear to 0.	1'b0
[4]	RW	cts changed (UART mode) This interrupt is generated when the cts signal changes. Software actively writes 1 to clear to 0.	1'b0
[3]	RW	rxfifo data timeout (UART&7816 mode) The data length in rxfifo is less than rxfifo trigger level but no data is received for N data cycles, An interrupt is generated.	1'b0

		Software actively writes 1 to clear to 0.	
[2]	RW	rxfifo trigger level interruptÿUART&7816 ÿÿÿ When the number of data in rxfifo changes from less than the number specified in rxfifo trigger level to greater than or equal to the number , this interrupt is generated. At this point, the current data frame size should be determined according to the rxfifo count. Software actively writes 1 to clear to 0.	1'b0
[1]	RW	txfifo trigger level interruptÿUART&7816 ÿÿÿ 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. Software actively writes 1 to clear to 0.	1'b0
[0]	RW	tx fifo empty interruptÿUART&7816 ÿÿÿ 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

18.5.9 FIFO Status Register

Table 157 UART&7816 FIFO Status Register

bit access		Instructions	reset value
[31:13]		Reserve	
[12]	RW	cts status (UART mode) the state of the current cts	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 1	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		The clock divider configures prescale. For example: apb_clk=40MHz prescale = 40 – 1 = 8'd39	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'b1: Timer generates an interrupt;	1'b0
		[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] RO	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] RO	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] RO	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] RO	Read the current count value of timer 6;	32’b0

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 LDOÿ

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.