MT793X IoT SDK for SDIO Slave User Guide

Version: 0.4
Release date: 2021-08-01

Version History

Version	Date	Description
0.1	2021-03-24	Initial Draft release
0.2	2021-03-31	Chapter 1: Introduction 1.1: delete unused feature 1.2: new section “Block Diagram” 1.6: add more description in Step 3,4,6 Chapter 2: Quick Question (New chapter) Chapter 6: SDIO Slave TX/Host RX 6.1: delete the unused field in GPD Chapter 7: SDIO Slave RX/Host TX 7.1: delete the unused field in GPD Chapter 11: APPENDIX 11.add new section “SDIO Slave CR”
0.3	2021-4-01	Chapter 1 1.6 : Add comment for every step to label host action or slave action Add note for step 5,6 Chapter 2: Answer for the meeting question. Chapter 6: SDIO Slave TX/Host RX 6.2: add more description and modify contents Chapter 7: SDIO Slave RX/Host TX 7.2: add more description and modify contents Delete Chapter 10: SDIO API API with its parameters will list in Doxygen files
0.4	2021-08-01	Add Hyperlink Modify description(add description and delete unsupported feature) List the tables and figures

Table of Contents

List of Figures

List of Tables

1 Introduction

The Secure Digital Input/Output (SDIO) card is based on and compatible with the SD (Secure Digital) memory card. The controller fully supports the SD memory card bus protocol as defined in SD Memory Card Specification Part 1 Physical Layer Specification version 2.0 and SDIO Specification version 2.0.

SDIO provides high-speed data IO with low power consumption. The SDIO module provides an SDIO2.0 card interface connected to the host and can support multiple speed modes including default speed mode and high speed mode.

1.1 Features

- SDIO 2.0 basic features
  - 1-bit and 4-bit SD data transfer modes
  - Default mode: Variable clock rate 0-25 MHz, up to 12.5 Mbps interface speed (using 4 parallel data lines)
  - High-Speed mode: Variable clock rate 0-50 MHz, up to 25 Mbps interface speed (using 4 data lines)
- CR and data port access
  - Supports control register (CR) port single read/write access (AHB slave)
  - Supports data port single and burst read/write access (AHB master)
  - Only support SDIO function 1
- DMA function
  - One Tx channel and two Rx channels
  - Moves Tx data from HIF buffer to SYSRAM, TCM
  - Moves Rx data or firmware prepared data from SYSRAM, TCM to HIF buffer

1.2 Block Diagram

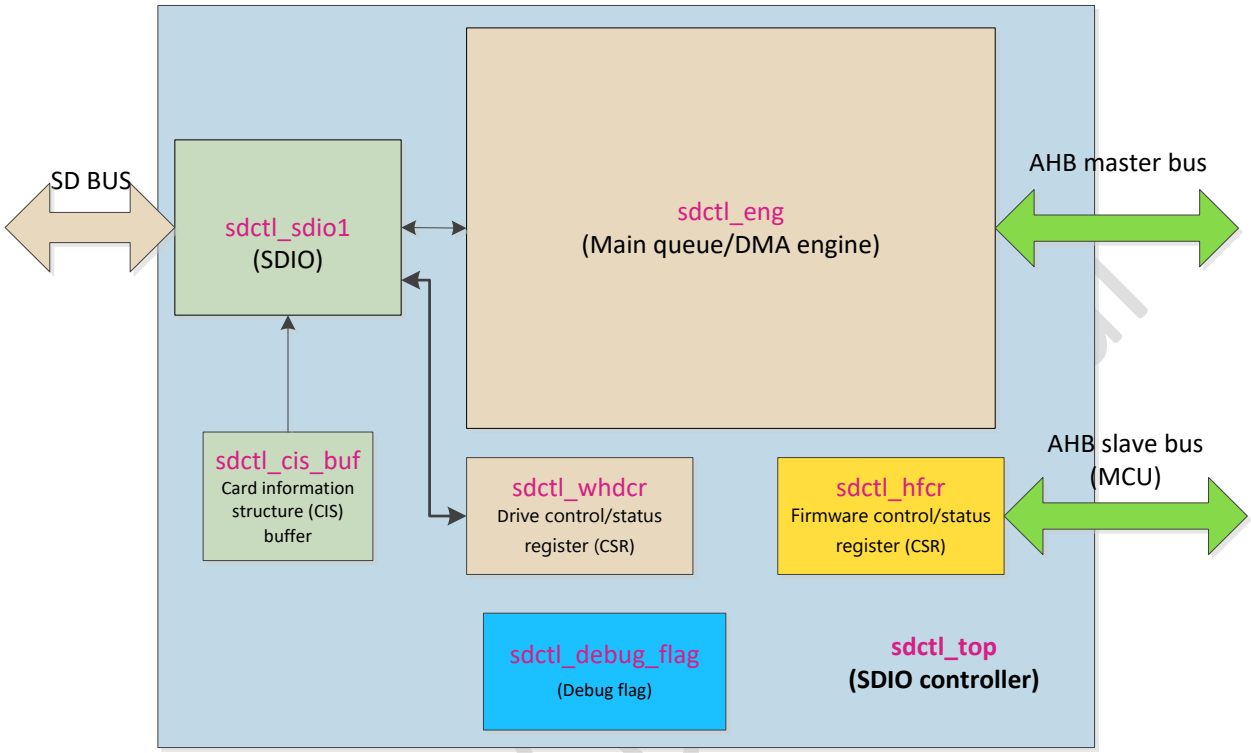

Figure 1 Block diagram

- The SD bus is connected to the SDIO host.
- The AHB master bus is for SDIO DMA to issue AHB transaction with address to get or give data
- The AHB slave bus is for MT793X MCU to access control CR of the slave module
- Sdctl_whdcr contains host domain control registers which can be accessed by host CMD53 or CMD52 only.

For probe procedure, it can be completed by sdctl_sdio1 and sdctl_cis_buf. It does not need the firmware to handle it.
Note that the AHB bus is ready when bootrom finishes.

1.3 Functions Description

From the external view, the SDIO interface mainly includes the SD bus and AHB master and slave. The AHB master is used for DMA operations and the AHB slave is used for register access from the MCU. The SD bus provides an interface for SD specification.

1.4 Signal Pins

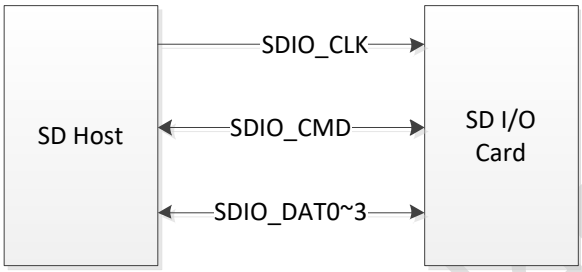

Figure 2 Signal connections to 4-bit SDIO cards

Table 1 SDIO pin definitions

Pin	Name	SD 4-bit mode		SD 1-bit mode	
1	SDIO_DAT3	DAT[3]	Data line3	N/C	Not used
2	SDIO_CMD	CMD	Command line	CMD	Command line
3	VSS1	VSS1	Ground	VSS1	Ground
4	VDD	VDD	Supply voltage	VDD	Supply voltage
5	SDIO_CLK	CLK	Clock	CLK	Clock
6	VSS2	VSS2	Ground	VSS2	Ground
7	SDIO_DAT0	DAT[0]	Data line0	DATA	Data line
8	SDIO_DAT1	DAT[1]	Data line1 or interrupt	IRQ	Interrupt
9	SDIO_DAT2	DAT[2]	Data line2	RW	Not used

1.5 SDIO Slave Pinmux

As shown in the following figure, the MT793x SDIO slave uses GPIO 6 to 11. Before performing any SDIO operation for the slave controller, GPIO 6 to 11 must be set to ALT 1. The default pinmux setting is ALT1.

Table 2 SDIO slave pinmux

GPIO	FUN	Description
GPIO 6	ALT fun 1	I0:SDIO_CLK
GPIO 7	ALT fun 1	B1:SDIO_CMD
GPIO 8	ALT fun 1	B1:SDIO_DAT0
GPIO 9	ALT fun 1	B1:SDIO_DAT1
GPIO 10	ALT fun 1	B1:SDIO_DAT2
GPIO 11	ALT fun 1	B1:SDIO_DAT3

1.6 SDIO Slave Firmware Flow Overview

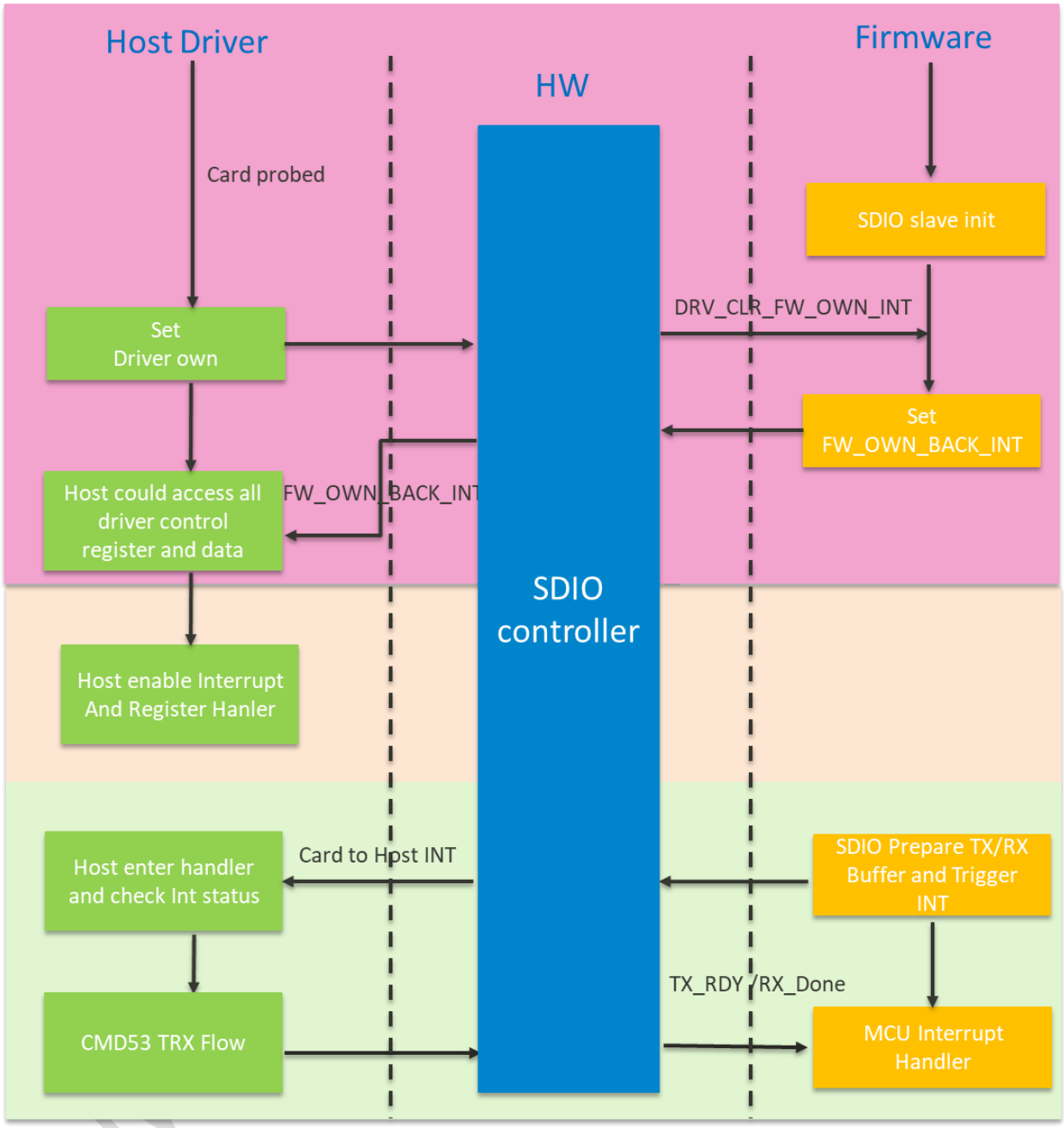

Figure 3 Procedure of SDIO slave TRX

This section describes a simple software flow of SDIO TX/RX

1. Slave: SDIO slave initialization

    1.1. Call API hal_sdio_slave_init(HAL_SDIO_SLAVE_PORT_0)

        1.1.1. SDIO slave hardware setting

        1.1.2. Clear SDIO slave queue and reset

        1.1.3. SDIO Slave Interrupt Setting

            1.1.3.1. Enable INT and decide which interrupt event can trigger the MT793X

            1.1.3.2. NVIC setting

    1.2 Register callback for INT: hal_sdio_slave_register_callback

2. Host: For the card probe procedure, refer to chapter 3.

3. Slave: SDIO Slave Driver own

To access all driver control register, the host should request driver own

    3.1. The host requests driver own bit.(Write **WHLPCR.** DRV_CLR_FW_OWN_by command 53)

    3.2. The firmware sets own back bit in the interrupt handler. (This is done by handler - sdio_slave_isr)

    3.3. The driver checks whether driver own is successful. (Read **WHLPCR.** W_FW_OWN_REQ_SET___by command53)

    See more information in chapter 4.

4. Host: For host enable Interrupt, refer to chapter 5.

    Call sdio_claim_irq(func, my_sdio_interrupt) to enable CCCR, which can be done before DRV_OWN

    Set **WHLPCR[0**] to 1, which can be done before DRV_OWN

    Set **WHIER** to 0x00000087, which must be done after DRV_OWN

5. Slave: The firmware prepares TRX buffer and trigger card to host INT

    5.1. Slave RX: call hal_sdio_slave_receive_dma

    5.2. Slave TX: call hal_sdio_slave_send_dma

    See more information in chapter 6 and chapter 7.

6. Host: The host enters interrupt handler and checks INT Status

After finishing Step 4, the host catches the corresponding card to host interrupt event, and runs into handler.

If the firmware calls hal_sdio_slave_receive_dma, **WHISR.**TX_DONE_INT_ is asserted.

    6.1. Check INT status bit by command53 to read **WHISR**.

    6.2. The host reads **WTSR0** to get packet number by command 53 before the host sends data.

6.3. The host writes data to **WTDR1** by command53; the slave receives data, and DMA pushes data to specific address automatically.

6.4. Once DMA completes the operation, TX1_RDY is asserted from the SDIO controller to MCU, and runs handler to clear status and the controller calls the callback registered by hal_sdio_slave_register_callback.

See more information in chapter 7.

Note 1: TX1_RDY means the host transmits data completely, and the MT793X can handle buffer received from the host. Please refer to 10.2 **HWFTE0SR** .

Note 2: Step 6.4 is performed by the firmware, not by the host.

If the firmware calls hal_sdio_slave_send_dma, **WHISR.** RX1_DONE_INT or RX0_DONE_INT is asserted.

6.5. Check INT status bit by command53 to read **WHISR**.

6.6. The host reads **WRPLR** to get packet length by command 53 before the host reads data.

6.7. The host reads data to **WRDR0** or **WRDR1** by command53; the slave sends data, and DMA gets data from specific address automatically

6.8. Once DMA completes the operation, RX0_DONE / RX1_DONE is asserted from the SDIO controller to MCU and the controller calls the callback registered by hal_sdio_slave_register_callback.

See more information in chapter 6.

Note 1: RX0_DONE / RX1_DONE means the host receives data completely, and the MT793X can clean the buffer that contains data sent from the host. Please refer to 10.2 . **HWFRE0SR .**

Note 2: Step 6.4 is performed by the firmware, not by the host

7. Repeat Step 5 and Step 6 for SDIO slave TX /RX

A brief description is provided in chapter 2 “How do the host and slave do flow control to TRX”

2 Quick Question

What is host domain CR?

A: Host domain CR means the control registers which can only be accessed by CMD52 and CMD53. CMD53 and CMD52 with a specific address can be used to communicate with the SDIO slave. Please refer to 10.1. Section 1.6 shows the flow for SDIO TRX.

What is GPD (general purpose descriptor) format?

A: The GPD format is a data structure that provides the SDIO controller the information about where it should take or get data from by DMA. The maximum data length in a GPD is limited to 4092 bytes.

What is DRV_OWN?

A: DRV_OWN is a mechanism for the host to get full control of the host domain CR. Refer to Section 1.6to check how it works.

What is the limitation of CMD53 TRX?

A: The data size requested by CMD53 with block mode must be the multiple of the block size. For the host TX, the host software should pad 0 at the end of data. For the host RX, padding is automatically done by the SDIO controller.

Can CMD52 read or write 4-byte data?

A: No, CMD52 does not use the DAT pin. Data is transferred by the CMD pin. CMD52 only reads or writes 8-bit data defined in SPEC.

S	D	Command Index 110100b	R/W flag	Function Number	RAW flag	Stuff	Register Address	Stuff	Write Data or Stuff Bits	CRC7	E
1	1	6	1	3	1	1	17	1	8	7	1

Figure 4 Format of CMD52

S	D	Command Index 110100b	Stuff	Response Flags Bit 7----------0	Read or Write Data	CRC7	E
1	1	6	16	8	8	7	1

Figure 5 Format of CMD52 response

How do the host and slave perform TRX flow control?

A: Refer to the following figure for details of the TRX flow.

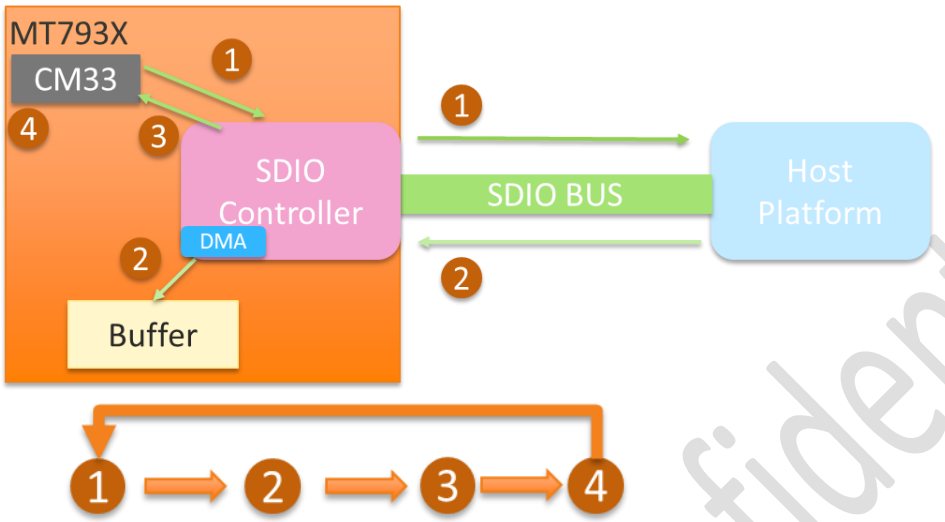

Figure 6 Mechanism of SDIO slave TRX

1. Call hal_sdio_slave_receive_dma or hal_sdio_slave_send_dma
   - ➔Set Destination or source address in GPD for DMA
   - ➔ Trigger card to Host Interrupt
2. Host run into handler and check interrupt status.
3. TX_DONE_INT asserted :
   - ➔ Following the step 6.2 & 6.3 in section 1.6
     step 6.3 will trigger DMA to get data from Buffer and push data to bus.

   RX0_DONE / RX1_DONE asserted:
   - ➔ Following the step 6.6 & 6.7 in section 1.6
     step 6.7 will trigger DMA to get data from bus and push data to buffer.
4. Once the DMA operations have been completed, SDIO controller triggers interrupt to CM33.
   - ➔Slave checked Status in sdio_slave_isr
   - ➔sdio_slave_isr will call your own callback which registered by hal_sdio_slave_register_callback
   - ➔your own callback may:
     1. Clearing the buffer, if RX0_DONE/RX1_DONE asserted. (Host CMD53 RX completed) Then doing 3.
     2. Manipulating data , if TX1_RDY asserted. (Host CMD53 TX completed) Then doing 3.
     3. Call hal_sdio_slave_receive_dma or hal_sdio_slave_send_dma to repeat TRX

3 SDIO Slave Probe Procedure

The SDIO slave hardware is ready as soon as the MT793X powers on (refer to 1.2). The default pinmux values of GPIO6 to 11 are also configured to the SDIO slave. The host can perform the probe sequence without firmware setting. Follow the standard procedure in Spec in order to complete the probe sequence successfully. The Linux kernel module has already implemented the sequence. When the card is inserted into the host slot, the host triggers Linux to run the probe procedure.

Once the probe procedure is complete, the block size is set to 256, which is the maximum block size of the MT793X, and the bus width is 4-bit by default in Linux.

4 SDIO Driver Own Procedure

When the host gets the driver own bit, it can get full control of the SDIO slave. The default status is firmware own. That it, the host must request the driver own bit first before data read or write. This chapter describes how to get the driver own bit.

After power on reset, the SDIO slave is controlled by the firmware and only limited registers are accessible normally; other registers are read with value 0 and written without function. When the host driver wants to get full control of the SDIO controller registers, it has to set FW_OWN_REQ_CLR to send an interrupt to the firmware. The firmware then sets FW_OWN_BACK_INT to give ownership to the host driver and informs the host that the host is ready to access all registers including data port.

The driver own flow is describe in Section 1.6.

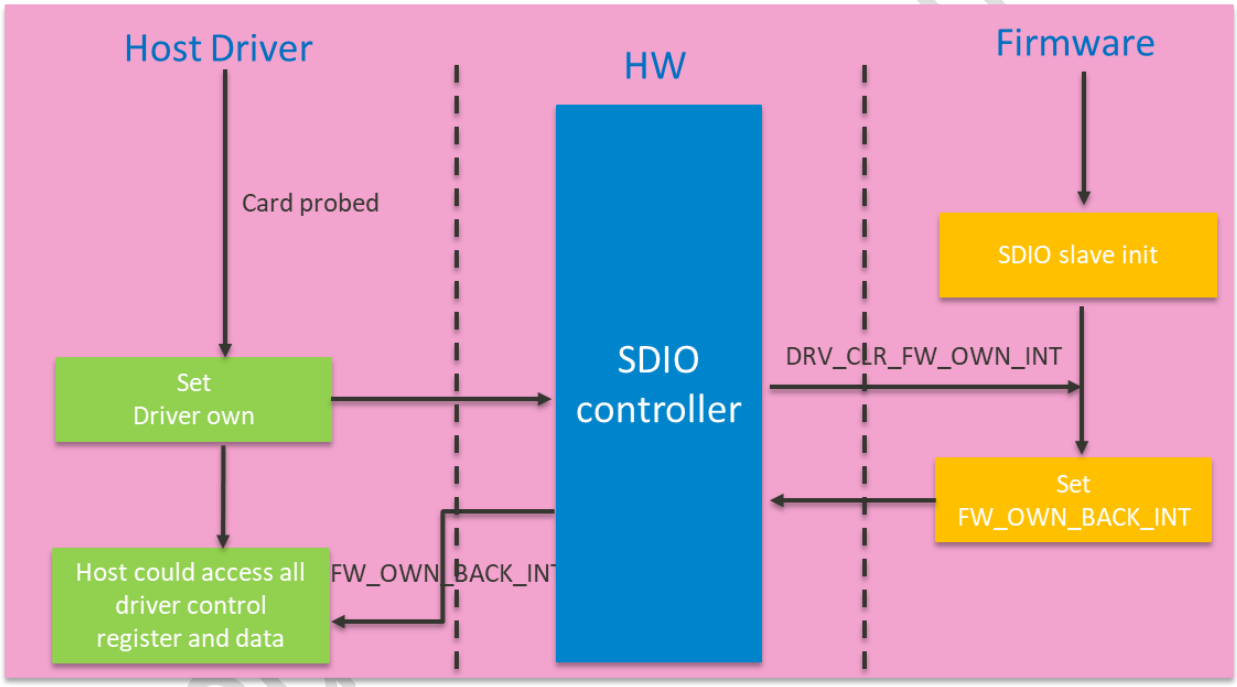

Figure 7 Procedure of Driver Own

5 SDIO Card to Host Interrupt

The SDIO slave controller can trigger interrupt to the host by DAT[1]. The host should set CCCR and write host domain CR to enable interrupt. After enabling interrupt, the host runs handler if it is registered.

Figure 8 Procedure of enabling card to host interrupt

5.1 Enable Host Interrupt

When the host wants to handle card to host interrupt, pay attention to the three types of registers introduced in the following sections.

5.1.1 CCCR IEN1

The Card Common Control Register (CCCR) allows quick host checking and provides control of enabling I/O card and interrupts on a per card (master) and per function basis. The host should set IEN1 to 1.

Table 3 CCCR IENx

Addr: Field	Type	Description
04h: IENx	R/W	Interrupt Enable for Function x. If this bit is cleared to 0, any interrupt from this function shall not be sent to the host. If this bit is set to 1, then this function's interrupt shall be sent to the host if the master Interrupt Enable (bit 0) is also set to 1

0x02	I/O Enable	IOE7	IOE6	IOE5	IOE4	IOE3	IOE2	IOE1	RFU
0x03	I/O Ready	IOR7	IOR6	IOR5	IOR4	IOR3	IOR2	IOR1	RFU
0x04	Int Enable	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IENM

Figure 9 CCCR

5.1.2 WHLPCR

When WHLPCR[0] is set, the card to host interrupt can be sent to the host. The meaning of bits of **WHLPCR** can refer to APPENDIX 10.1.

5.1.3 WHIER

WHIER controls which interrupt event can trigger interrupt from the slave controller to the host. If the host does not set WHIER, the corresponding event does not generate interrupt to the host. The meaning of bits of WHIER can be in APPENDIX 10.1.

Note: All of the above registers should be set, if the host wants to receive card to host interrupt.

6 SDIO Slave TX/Host RX

The software is expected to issue sufficient length of blocks/bytes to receive the RX packet(s) queuing in the SDIO controller, with the RX-packet length reported by either RX port read. The SDIO controller pads 0 to the remaining part of the command data after all expected RX packets are received by the host.

If the host driver knows that there is RX packet data to be received, it could issue a CMD53 with byte mode or block mode RX command to receive all RX data whose packet length has been read. Otherwise, it could wait for interrupt RX_DONE_INT to read **WRPLR** to get RX packet number and all the RX packet lengths, and then issues a CMD53 to read port for reading data from the slave.

Note that the whole RX packet should be received within a transaction by one CMD53. The packet can not be cut to be transmitted by two or more commands. In normal mode, the host should reserve enough space for all RX packet data with RX lengths reported in the previous length reading. The maximum length of data is 4092.

The legal format for command 53 Read Data is shown in 6.2.

6.1 SDIO Slave TX / Host RX Flow

1. Slave TX:

Prepare slave TX buffer and trigger INT

➔ Call hal_sdio_slave_send_dma

2. Host RX:

The host enters handler and checks INT Status

➔ Check INT status bit by command53 to read WHISR (WHISR. RX1_DONE_INT or RX0_DONE_INT will be asserted)

CMD53 RX Flow

➔ The host reads packet length by command 53 before the host receives data

Packet length can be obtained from reading WRPLR

➔ The host reads data to WRDR0 or WRDR1 by command53; the slave sends data, and DMA gets data from specific address automatically

➔ Once DMA completes the operation, RX0_DONE / RX1_DONE is asserted from the SDIO controller to MCU and the controller calls the callback registered by hal_sdio_slave_register_callback.

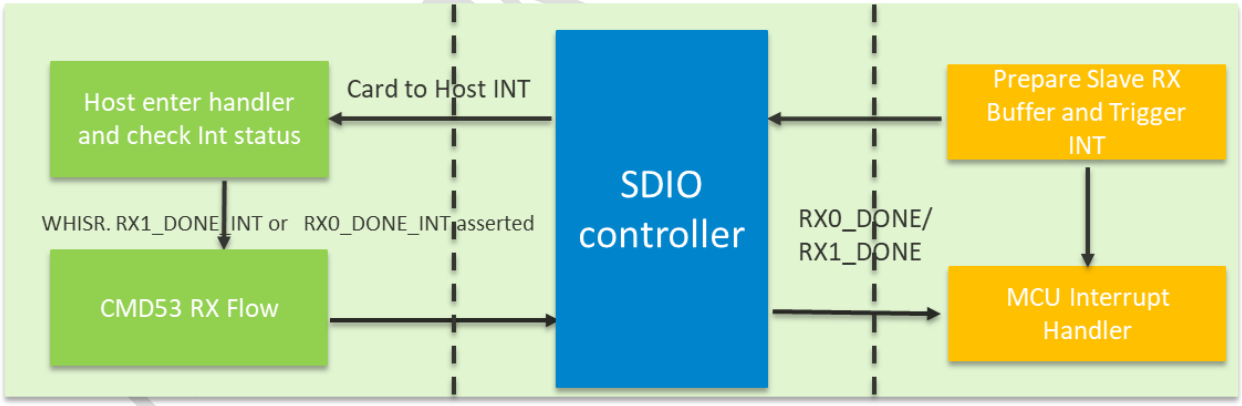

Figure 10 Procedure of SDIO slave TX

Note: RX0_DONE / RX1_DONE means the host receives data completely, and the MT793X can clean the buffer that contains data sent from the host. Please refer to 10.2 . "**HWFRE0SR**"

6.2 Command 53 Host RX Packet Format

6.2.1 Byte Mode

As mentioned above, the whole RX packet must be received within a transaction by one CMD53 with byte mode. When calling CMD53, the SDIO controller copies data from the buffer to queue by DMA and sends data to the host.

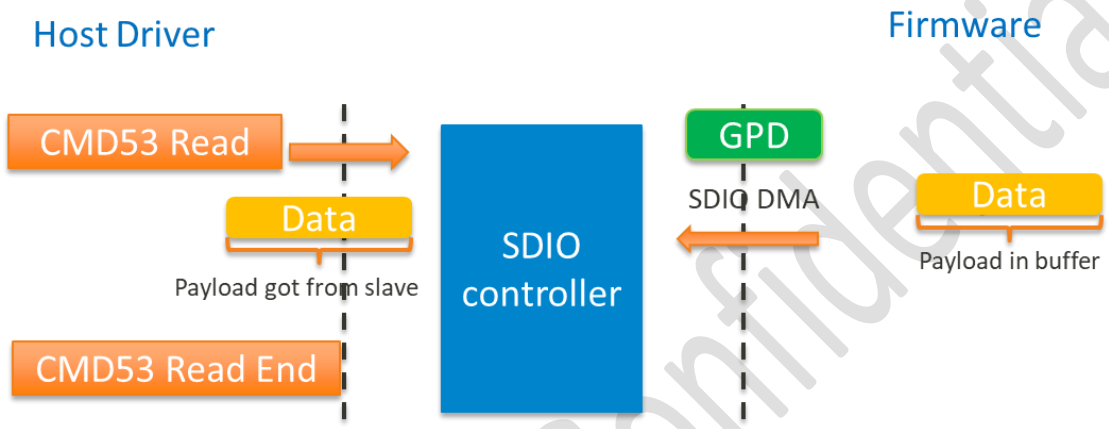

Figure 11 SDIO host RX byte mode

As mentioned above, the whole RX packet must be received within a transaction by one CMD53. Due to hardware limitation, after the data size is requested by the host via CMD53,if the data size is not the multiple of the block size, the data should be padded to 0. Padding 0 is automatically done by the SDIO controller.

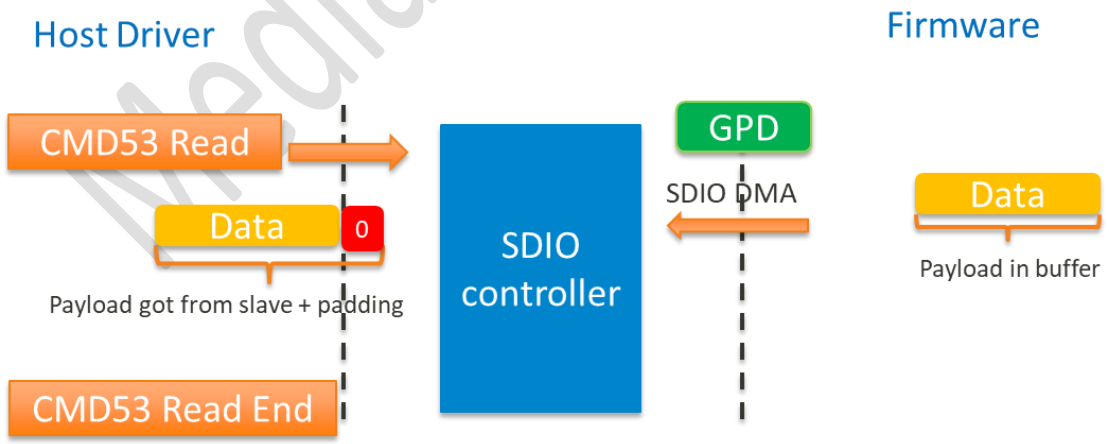

Figure 12 SDIO host RX block mode

Note that each packet should not be cut within transaction boundary. To achieve this, padding is needed if the (concatenated) packet length cannot fit into a complete SDIO data transition.

7 SDIO Slave RX/Host TX

If the host driver knows that there are available buffers for transmission, it could issue a CMD53 with byte mode or block mode TX command to transmit all data. It could wait for interrupt TX_DONE_INT to read **WTSR0** to get TX packet count, and then issues CMD53 with TX command. Note that the whole TX packet should be transmitted within a transaction by one CMD53. The packet can not be cut to be transmitted by two or more commands. Therefore, the host driver must pad 0 at the end of data to make the packet to be the multiple of block size so that the data can be transmitted in one command.

Padding at the tail of the packet is used to inform the SDIO controller of the end of the packet. The header, placed at the beginning of the packet, is 4-byte data that describes the size of a packet. The maximum length of data is 4092. The length information in header in each TX packet is used to inform the SDIO controller if the data transfer is completed.

The legal format for command53 write Data is shown in 7.2.

7.1 SDIO Slave RX / Host TX Flow

1. Slave RX:

   Prepare slave RX buffer and trigger INT

   ➔ Call hal_sdio_slave_receive_dma

2. Host TX:

The host enters handler and checks INT Status

➔ Check INT status bit by command53 to read WHISR (WHISR.TX_DONE_INT will be asserted)

CMD53 TX Flow

➔ The host reads packet number by command 53 before the host sends data

Packet number can be obtained from reading WTSR0

➔ The host writes data to WTDR1 by command53, the slave receives data, and DMA pushes data to specific address automatically.

➔ Once DMA completes the operation, TX1_RDY is asserted from the SDIO controller to MCU, and runs handler to clear status and the controller calls the callback registered by hal_sdio_slave_register_callback.

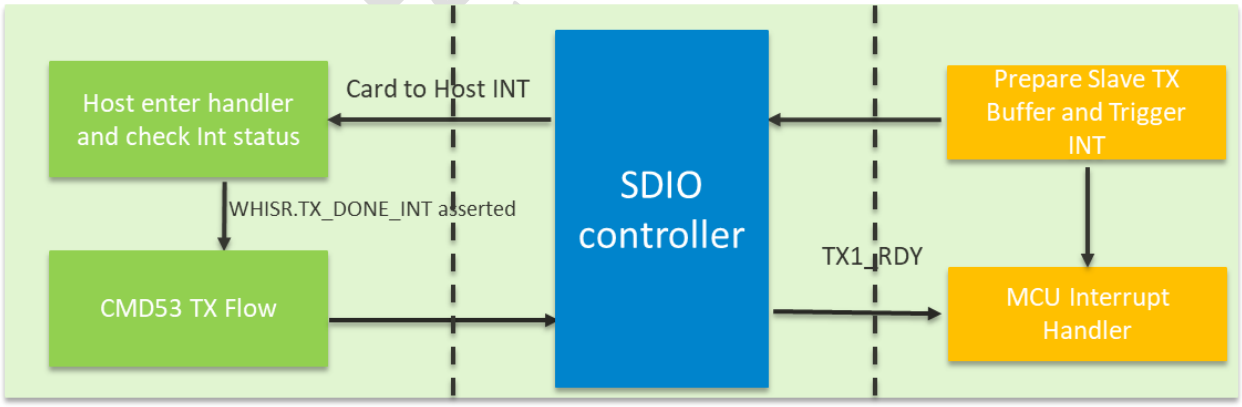

Figure 13 Procedure of SDIO Slave RX

Note: TX1_RDY means the host transmits data completely, and the MT793X can handle buffer received from the host. Please refer to 10.2 "HWFTE0SR **"**.

7.2 Command 53 Host TX Packet Format

7.2.1 Byte Mode

As mentioned above, the whole TX packet must be sent within a transaction by one CMD53 with byte mode. When calling CMD53, the host pushes data to queue and the SDIO controller DMA sends data to buffer. The destination address of DMA is described in GPD. The host software adds a 4-byte header at the beginning of the data that describes data length.

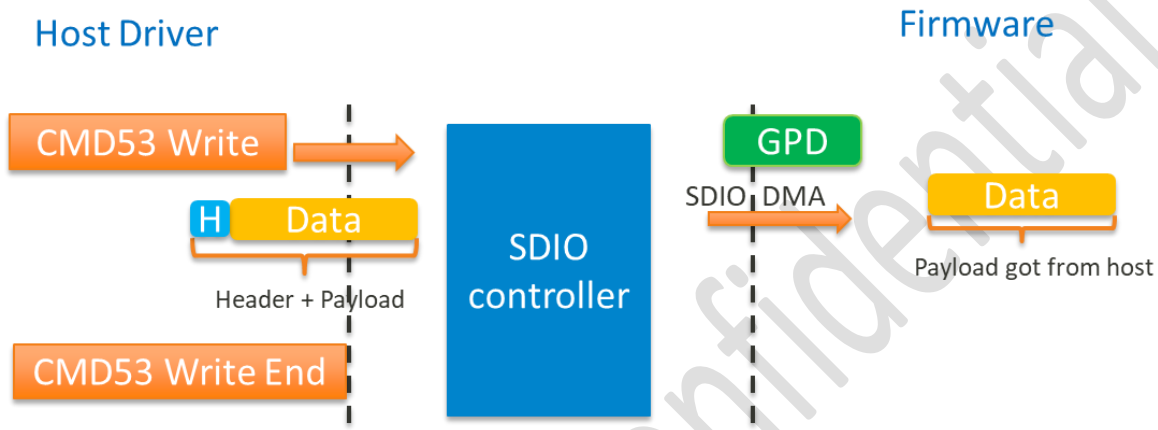

Figure 14 SDIO host TX byte mode

7.2.2 Block Mode

The whole RX packet must be sent within a transaction by one CMD53. Due to hardware limitation, after the data size is requested by the host via CMD53, if the data size is not the multiple of block size, the data is padded to 0. The host software should pad 0 at the end of data and add a 4-byte header at the beginning of data that describes data length.

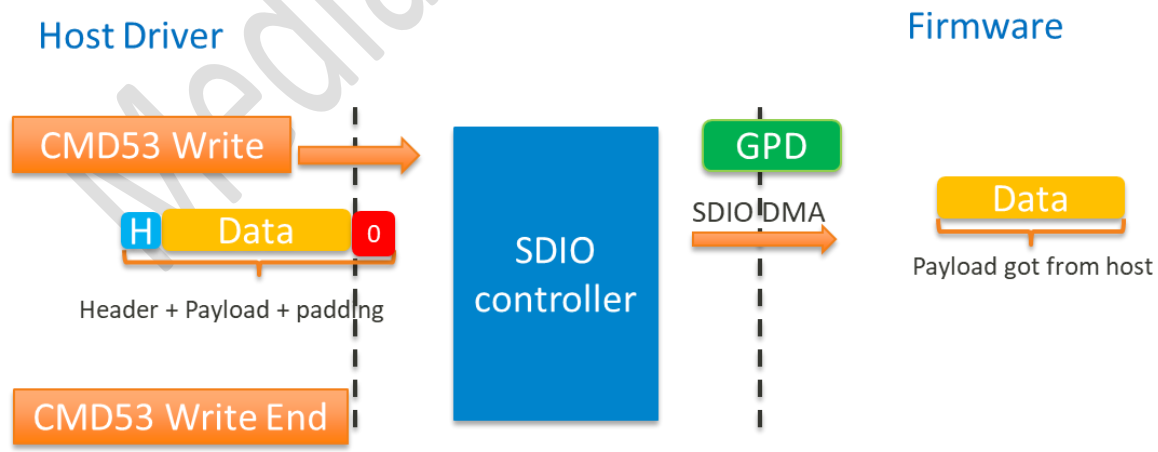

Figure 15 SDIO host TX block mode

8 SDIO Mailbox

The mailbox mechanism is used for firmware and host driver communication. However, the mailbox also needs to coordinate with software interrupt to complete the mechanism. First, the host driver or firmware sets the appropriate mailbox content and then sets the software interrupt to inform the opposite. Whenever the opposite receives the interrupt and finds the interrupt source is software interrupt, it checks the mailbox.

There are two mailboxes and they are unidirectional. The way the mailbox works is similar to handshake and depends on the software usage.

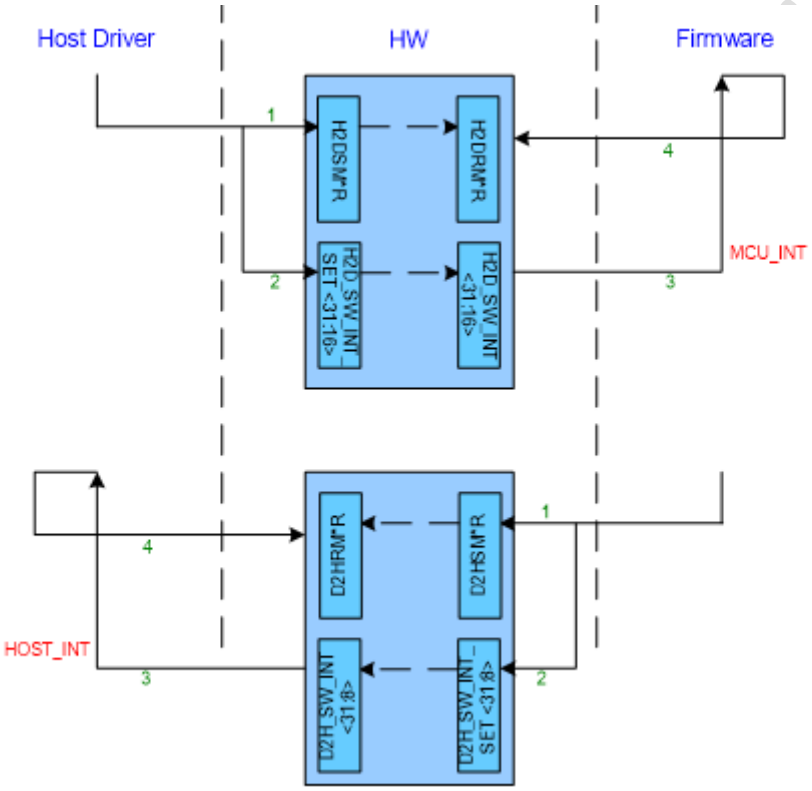

Figure 16 Mailbox

9 SDIO Slave Firmware Own

If the host does not want to access the SDIO controller, it could give ownership to the firmware by setting FW_OWN_REQ_SET to inform the firmware. If there is no other task to executed, the firmware could disable all unnecessary clocks and go to sleep mode. If the host wants to wake up the SDIO controller, it could set FW_OWN_REQ_CLR as specified in Chapter 3.

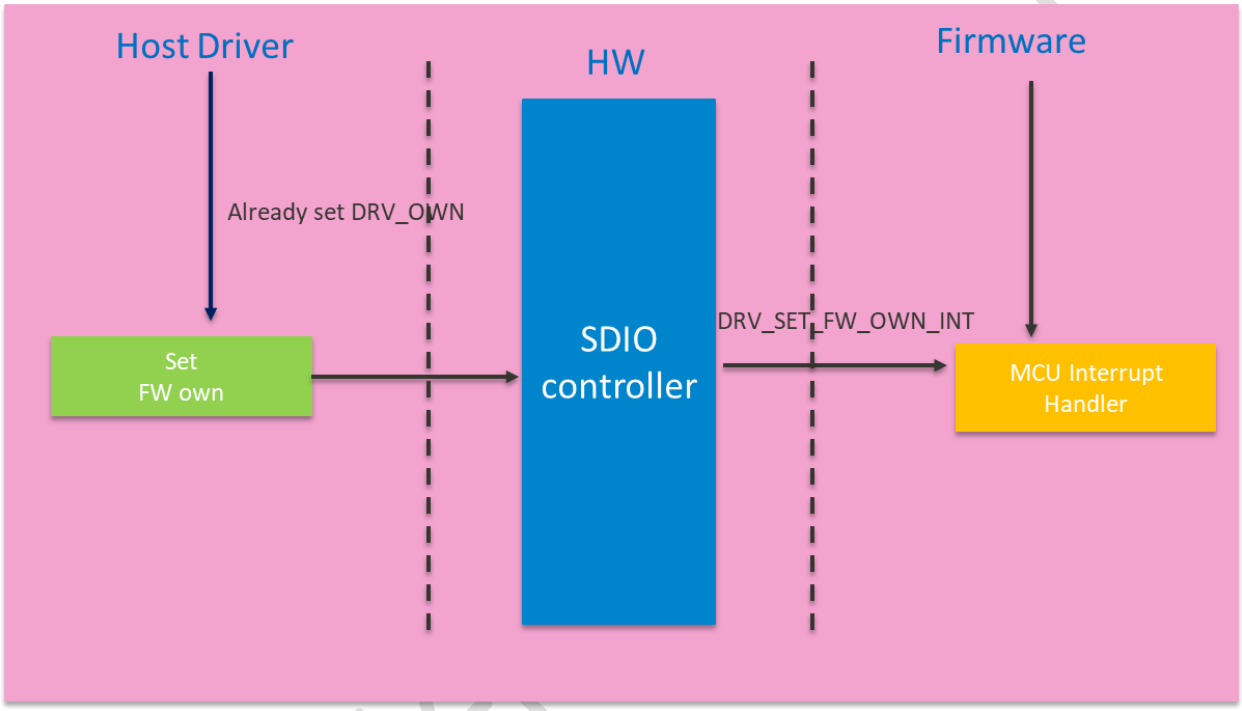

Figure 17 Procedure of SDIO slave firmware own

9.1 Firmware Own Flow

The host driver writes **WHLPCR.** W_FW_OWN_REQ_SET by CMD53 to set firmware own bit.

10 APPENDIX

10.1 Host Domain CR

Module name: SDIO_SLV Base address: (+0x0000000)

Address	Name	Width	Register Function
00000000	WCIR	32	WLAN Chip ID Register
00000004	WHLPCR	32	WLAN HIF Low Power Control Register
0000000C	WHCR	32	WLAN HIF Control Register
00000010	WHISR	32	WLAN HIF Interrupt Status Register
00000014	WHIER	32	WLAN HIF Interrupt Enable Register
00000020	WASR	32	WLAN Abnormal Status Register
00000024	WSICR	32	WLAN Software Interrupt Control Register
00000028	WTSR0	32	WLAN TX Status Register
0000002C	WTSR1	32	WLAN TX Status Register
00000034	WTDR1	32	WLAN TX Data Register 1
00000050	WRDR0	32	WLAN RX Data Register 0
00000054	WRDR1	32	WLAN RX Data Register 1
00000070	H2DSM0R	32	Host to Device Send Mailbox 0 Register
00000074	H2DSM1R	32	Host to Device Send Mailbox 1 Register
00000078	D2HRM0R	32	Device to Host Receive Mailbox 0 Register
0000007C	D2HRM1R	32	Device to Host Receive Mailbox 1 Register
00000090	WRPLR	32	WLAN RX Packet Length Register

00000000 WCIR WLAN Chip ID Register 00106630

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DEVICE_STATUS										W_FUNC_RDY	POR_INDICATOR	REVISION_ID			
Type	RO										RO	W1C	RO			
Reset	0	0	0	0	0	0	0	0			0	1	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CHIP_ID															
Type	RO															
Reset	0	1	1	0	0	1	1	0	0	0	1	1	0	0	0	0

Bit(s)	Name	Description
31:24	DEVICE_STATUS	**These status bits are defined by users and could be read by host driver via SDIO bus interface. For example, watch dog reset status could be one that could be read by host driver even when there is no AHB clock in SDIO controller.**
21	W_FUNC_RDY	**Indicate that WLAN functional block has finished its initial procedure and it is ready for normal operation._x000D_** This is a sticky bit of HWFCR.W_FUNC_RDY._x000D_

Bit(s)	Name	Description
		Driver will keep polling this bit on initialization. Once after FW is ready and set corresponding bit in FW, driver can do following control to FW._x000D_ 0: WLAN functional block is not ready for normal operation. 1: WLAN functional block is ready for normal operation._x000D_
20	POR_INDICATOR	**This bit indicates a reset occurs including external pin reset, power detect reset, power on reset, SDIO CCCR(0x06).Bit[3] reset (only in SDIO). _x000D_** Write 1 to clear this bit. Write 0 is meaningless._x000D_
19:16	REVISION_ID	**Revision ID_x000D_**
15:0	CHIP_ID	**Chip ID_x000D_**

00000004 WHLPCR WLAN HIF Low Power Control Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							W_FW_OWN_REQ_CLR	W_FW_OWN_REQ_SET							W_INT_EN_CLR	W_INT_EN_SET
Type							W1S	W1S							W1S	W1S
Reset							0	0							0	0

Bit(s)	Name	Description
9	W_FW_OWN_REQ_CLR	**Write 1 to this bit to request firmware to return the ownership of chip WLAN function to host driver. Write 0 has no meaning (Refer chapter "Power management" for details)._x000D_** Read always return 0._x000D_
8	W_FW_OWN_REQ_SET	This bit will be set on initial, or by firmware written 1 to HWFICR. FW_OWN_BACK_INT_SET or any driver-domain WLAN interrupts. **Write 1 to this bit to transfer ownership of chip WLAN function to firmware. Write 0 has no meaning (Refer chapter "Power management" for details). Host driver should set this bit to give ownership to firmware only when host driver has ownership. _x000D_** Read will get the status of WLAN_DRV_OWN._x000D_ WLAN_DRV_OWN indicates that software driver has the ownership of chip WLAN sub-system._x000D_ 0: WLAN driver doesn't have ownership 1: WLAN driver has ownership_x000D_
1	W_INT_EN_CLR	**Write 0 has no meaning, and write 1 to clear WLAN interrupt enable signal._x000D_** Read always return 0._x000D_
0	W_INT_EN_SET	**Write 0 has no meaning, and write 1 to set WLAN interrupt enable signal._x000D_** Read will get the status of W_INT_EN._x000D_

Bit(s)	Name	Description
		_x000D_ W_INT_EN indicates the current value of WLAN interrupt enable signal. This enable signal is used for controlling the output of WLAN interrupt signal._x000D_ 0: WLAN interrupt can't output to host. 1: WLAN interrupt can output to host._x000D_

0000000C **WHCR** **WLAN HIF Control Register** **00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																RX_ENHANCE_MODE
Type																RW
Reset																0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name			MAX_HIF_RX_LEN_NUM										RPT_OWN_RX_PACKET_LEN	RECV_MAILBOX_RD_CLR_EN	W_INT_CLR_CTRL	
Type			RW										RW	RW	RW	
Reset			0	0	0	0	0	0					0	0	0	

Bit(s)	Name	Description
16	RX_ENHANCE_MODE	**Enable the read of TX count status, RX length, and mailbox information in RX packet enhance mode. Refer chapter 3.4 for more details._x000D_** 0: disable RX packet enhance mode 1: Enable the read of TX count status, RX length, and mailbox information in RX packet enhance mode._x000D_
13:8	MAX_HIF_RX_LEN_NUM	**The maximum number of SDIO controller to report the per-queue RX packets length via INT/ RX enhance mode._x000D_** 0: report entire 64 RX packets length in the same RX queue without limitation. Others (N): report at most N RX packet lengths for each RX queue
3	RPT_OWN_RX_PACKET_LEN	**This field is to control the rx packet report length and structure during enhance mode. If this bit is set to 1, each rx queue can report its own length according to the setting in WPLRCR. Also, the total report length would be changed if this bit is set. Host driver should parse the enhance mode status according to the length setting in WPLRCR to get correct information.** 0: disable the function that each rx queue can report its own packet length and the maximal report length is constrained by max_hif_rx_len_num field in WHCR 1: enable the function that each rx queue can report its own packet length and the maximal report length can be different by each queue according to the setting in WPLRCR
2	RECV_MAILBOX_RD_CLR_EN	**This is to control whether the received mail-box (D2HRM0R, D2HRM1R) will be read cleared or not (this include read from enhance mode structure)._x000D_**

Bit(s)	Name	Description
1	W_INT_CLR_CTRL	1: read clear 0: no effect after read_x000D_ **This bit is used to select the clear mechanism of WLAN interrupt statue (WHISR)._x000D_** 0: Read clear 1: Write 1 clear_x000D_

00000010 WHISR WLAN HIF Interrupt Status Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	D2H_SW_INT															
Type	RC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	D2H_SW_INT								FW_OWN_BACK_INT	ABNORMAL_INT		RX3_DONE_INT	RX2_DONE_INT	RX1_DONE_INT	RX0_DONE_INT	TX_DONE_INT
Type	RC								RC	RO		RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0

Bit(s)	Name	Description
31:8	D2H_SW_INT	**This field is used for software interrupt for WLAN function in FW to host driver direction._x000D_** WLAN firmware writes 1s to HWFICR.Bit[31:8] will set corresponding bit field._x000D_
7	FW_OWN_BACK_INT	**Firmware has returned the ownership to host driver. This field is set with driver own-back only.** Firmware write 1 to HWFICR. Bit[4] will set this bit.
6	ABNORMAL_INT	**Abnormal event interrupt._x000D_** The abnormal status will be shown in WASR, which includes_x000D_ Data overflow of WLAN TX0 and TX1 port._x000D_ Data underflow of WLAN RX0 and RX1 port._x000D_ FW_OWN_INVALID_ACCESS_x000D_
4	RX3_DONE_INT	**When any of the RX length data of RX3 is existed in HIF RX length FIFO, this bit will be asserted._x000D_**
3	RX2_DONE_INT	**When any of the RX length data of RX2 is existed in HIF RX length FIFO, this bit will be asserted._x000D_**
2	RX1_DONE_INT	**When any of the RX length data of RX1 is existed in HIF RX length FIFO, this bit will be asserted._x000D_**
1	RX0_DONE_INT	**When any of the RX length data of RX0 is existed in HIF RX length FIFO, this bit will be asserted._x000D_**
0	TX_DONE_INT	**If WTSR0 or WTSR1 is not 0, this bit will be set._x000D_** 0: WTSR0 and WTSR1 is 0. 1: WTSR0 or WTSR1 is not 0._x000D_

00000014 WHIER WLAN HIF Interrupt Enable Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Name	D2H_SW_INT_EN															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	D2H_SW_INT_EN								FW_OWN_BACK_INT_EN	ABNORMAL_INT_EN		RX3_DONE_INT_EN	RX2_DONE_INT_EN	RX1_DONE_INT_EN	RX0_DONE_INT_EN	TX_DONE_INT_EN
Type	RW								RW	RW		RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0

Bit(s)	Name	Description
31:8	D2H_SW_INT_EN	**WLAN host interrupt output control bits._x000D_** If any bit is_x000D_ 0: Mask the WLAN related bit interrupt output, corresponding bits will be still written to WHISR without triggering interrupt. 1: Enable the WLAN related bit interrupt output._x000D_
7	FW_OWN_BACK_INT_EN	**WLAN host interrupt output control bits._x000D_** If any bit is_x000D_ 0: Mask the WLAN related bit interrupt output, corresponding bits will be still written to WHISR without triggering interrupt. 1: Enable the WLAN related bit interrupt output._x000D_
6	ABNORMAL_INT_EN	**WLAN host interrupt output control bits._x000D_** If any bit is_x000D_ 0: Mask the WLAN related bit interrupt output, corresponding bits will be still written to WHISR without triggering interrupt. 1: Enable the WLAN related bit interrupt output._x000D_
4	RX3_DONE_INT_EN	**WLAN host interrupt output control bits._x000D_** If any bit is_x000D_ 0: Mask the WLAN related bit interrupt output, corresponding bits will be still written to WHISR without triggering interrupt. 1: Enable the WLAN related bit interrupt output._x000D_
3	RX2_DONE_INT_EN	**WLAN host interrupt output control bits._x000D_** If any bit is_x000D_ 0: Mask the WLAN related bit interrupt output, corresponding bits will be still written to WHISR without triggering interrupt. 1: Enable the WLAN related bit interrupt output._x000D_
2	RX1_DONE_INT_EN	**WLAN host interrupt output control bits._x000D_** If any bit is_x000D_ 0: Mask the WLAN related bit interrupt output, corresponding bits will be still written to WHISR without triggering interrupt. 1: Enable the WLAN related bit interrupt output._x000D_
1	RX0_DONE_INT_EN	**WLAN host interrupt output control bits._x000D_** If any bit is_x000D_ 0: Mask the WLAN related bit interrupt output, corresponding bits will be still written to WHISR without triggering interrupt. 1: Enable the WLAN related bit interrupt output._x000D_
0	TX_DONE_INT_EN	**WLAN host interrupt output control bits._x000D_** If any bit is_x000D_ 0: Mask the WLAN related bit interrupt output, corresponding bits will be still written to WHISR without triggering interrupt. 1: Enable the WLAN related bit interrupt output._x000D_

00000020 WASR WLAN Abnormal Status Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																FW_OWN_INVALID_ACCESS
Type																RC
Reset																0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					RX3_UNDERFLOW	RX2_UNDERFLOW	RX1_UNDERFLOW	RX0_UNDERFLOW							TX1_OVERFLOW	TX0_OVERFLOW
Type					RC	RC	RC	RC							RC	RC
Reset					0	0	0	0							0	0

Bit(s)	Name	Description
16	FW_OWN_INVALID_ACCESS	**It will be asserted when register other than WCIR, WHLPCR, WSPICSR, WSDIOCSR, WEHPICSR, and firmware download relative registers are accessed when FW own = 1** It is purely for host driver debug purpose.
11	RX3_UNDERFLOW	**Data underflow of WLAN RX3 port._x000D_**
10	RX2_UNDERFLOW	**Data underflow of WLAN RX2 port._x000D_**
9	RX1_UNDERFLOW	**Data underflow of WLAN RX1 port._x000D_**
8	RX0_UNDERFLOW	**Data underflow of WLAN RX0 port._x000D_**
1	TX1_OVERFLOW	**Data overflow of WLAN TX1 port._x000D_**
0	TX0_OVERFLOW	**Data overflow of WLAN TX0 port._x000D_**

00000024 WSICR WLAN Software Interrupt Control Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	H2D_SW_INT_SET															
Type	W1S															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																
Type																
Reset																

Bit(s)	Name	Description
31:16	H2D_SW_INT_SET	**Host driver writes 1s will set HWFISR. HOST_DRIVER_INT. Write 0 is meaningless.** Read always return 0. This is used as a communication between FW to driver, with interrupt functionality to SDIO controller.

00000028 WTSR0 WLAN TX Status Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TQ3_CNT								TQ2_CNT							
Type	RC								RC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TQ1_CNT								TQ0_CNT							
Type	RC								RC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:24	TQ3_CNT	**This field indicates the released packet count of TQ3 during two WTSR0 read access._x000D_** This field is cleared by read operation. Write has no meaning._x000D_
23:16	TQ2_CNT	**This field indicates the released packet count of TQ2 during two WTSR0 read access._x000D_** This field is cleared by read operation. Write has no meaning._x000D_
15:8	TQ1_CNT	**This field indicates the released packet count of TQ1 during two WTSR0 read access._x000D_** This field is cleared by read operation. Write has no meaning._x000D_
7:0	TQ0_CNT	**This field indicates the released packet count of TQ0 during two WTSR0 read access._x000D_** This field is cleared by read operation. Write has no meaning._x000D_

0000002C **WTSR1** **WLAN TX Status Register** **00000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TQ7_CNT								TQ6_CNT							
Type	RC								RC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TQ5_CNT								TQ4_CNT							
Type	RC								RC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:24	TQ7_CNT	**This field indicates the released packet count of TQ7 during two WTSR0 read access._x000D_** This field is cleared by read operation. Write has no meaning._x000D_
23:16	TQ6_CNT	**This field indicates the released packet count of TQ6 during two WTSR0 read access._x000D_** This field is cleared by read operation. Write has no meaning._x000D_
15:8	TQ5_CNT	**This field indicates the released packet count of TQ5 during two WTSR1 read access._x000D_** This field is cleared by read operation. Write has no meaning._x000D_
7:0	TQ4_CNT	**This field indicates the released packet count of TQ4 during two WTSR1 read access._x000D_** This field is cleared by read operation. Write has no meaning._x000D_

00000034 WTDR1 WLAN TX Data Register 1 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TX1_DATA															
Type	WO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TX1_DATA															
Type	WO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	TX1_DATA	**TX1 write data port. Read always return 0._x000D_** _x000D_ Data must be padded to multiples of block when the data to write is more than the size of a single block. Writing data with multiple blocks in one transaction and the remaining in another with byte mode is prohibited._x000D_

00000050 WRDR0 WLAN RX Data Register 0 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RX0_DATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RX0_DATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	RX0_DATA	**RX0 read data port. Write has no effect._x000D_** The RX0 data port support data aggregation. Driver should read the entire RX packets by last SDIO controller indicated information. The number of total RX aggregation packets is restricted by WHCR. MAX_HIF_RX_LEN_NUM._x000D_ (details is in chapter 3.1):_x000D_ _x000D_ Length to read must be extended to multiples of block when the data to read is more than the size of a single block. Reading data with multiple blocks in one transaction and the remaining in another with byte mode is prohibited._x000D_ _x000D_ Also as long as host driver knows the total available packet number and length via enhanced interrupt response and/or RX packet enhanced mode, host driver must read all RX packets in a single transaction. Reading for partial packets is prohibited either._x000D_

Bit(s)	Name	Description

00000054 WRDR1 WLAN RX Data Register 1 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RX1_DATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RX1_DATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	RX1_DATA	**RX1 read data port. Write has no effect._x000D_** The RX1 data port support data aggregation. Driver should read the entire RX packets by last SDIO controller indicated information. The number of total RX aggregation packets is restricted by WHCR. MAX_HIF_RX_LEN_NUM._x000D_ (details is in chapter 3.1):_x000D_ _x000D_ Data length to read must be extended to multiples of block when the data to read is more than the size of a single block. Reading data with multiple blocks in one transaction and the remaining in another with byte mode is prohibited._x000D_ _x000D_ Also as long as host driver knows the total available packet number and length via enhanced interrupt response and/or RX packet enhanced mode, host driver must read all RX packets in a single transaction. Reading for partial packets is prohibited either._x000D_

00000070 H2DSM0R Host to Device Send Mailbox 0 Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	H2D_SM0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	H2D_SM0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	H2D_SM0	**This register is used by host driver to transmit data to SDIO controller, which will be updated to H2DRM0R and read by FW. _x000D_**

00000074 H2DSM1R Host to Device Send Mailbox 1 Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	H2D_SM1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	H2D_SM1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	H2D_SM1	**This register is used by host driver to transmit data to SDIO controller, which will be updated to H2DRM1R and read by FW. _x000D_**

00000078 D2HRM0R Device to Host Receive Mailbox 0 Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	D2H_RM0															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	D2H_RM0															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	D2H_RM0	**This register is used by host driver to receive data from SDIO controller, which is updated through D2HSM0R by FW._x000D_** The property of RO/ RC is by control of WHCR. RECV_MAILBOX_RD_CLR_EN bit._x000D_

0000007C D2HRM1R Device to Host Receive Mailbox 1 Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	D2H_RM1															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	D2H_RM1															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	D2H_RM1	**This register is used by host driver to receive data from SDIO controller, which is updated through D2HSM1R by FW._x000D_** The property of RO/ RC is by control of WHCR. RECV_MAILBOX_RD_CLR_EN bit._x000D_

00000080 D2HRM2R Device to Host Receive Mailbox 2 Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	D2H_RM2															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	D2H_RM2															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	D2H_RM2	**This register is used by host driver to receive data from SDIO controller, which is updated through D2HSM2R by FW. For synchronization of hardware, it is recommended to read this register more than once to give more host clock cycles to device to get the latest result, especially for EHPI interface._x000D_** Note that this register could be read when there is no AHB clock, i.e. host driver could get this message when chip is in low power mode._x000D_

00000090 WRPLR WLAN RX Packet Length Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RX1_PACKET_LENGTH															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RX0_PACKET_LENGTH															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:16	RX1_PACKET_LENGTH	**This register is used to get the next RX packet length in the RX1 length FIFO, which is updated by FW HWRQ1CR._x000D_** When this field is read, it will report only 1 RX packet length in this RX queue, and at most 1 packet will return by next RX port read._x000D_
15:0	RX0_PACKET_LENGTH	**This register is used to get the next RX packet length in the RX0 length FIFO, which is updated by FW HWRQ0CR._x000D_** When this field is read, it will report only 1 RX packet length in this RX queue, and at most 1 packet will return by next RX port read._x000D_

10.2 SDIO Slave CR

Address	Name	Width	Register Function
38130000	HGFCR	32	HIF Global Firmware Configuration Register
38130004	HGFISR	32	HIF Global Firmware Interrupt Status Register
38130008	HGFIER	32	HIF Global Firmware Interrupt Enable Register
3813001C	HGH2DR	32	HIF Global Host to Device Register
38130100	HWFISR	32	HIF WLAN Firmware Interrupt Status Register
38130104	HWFIER	32	HIF WLAN Firmware Interrupt Enable Register
38130110	HWFTE0SR	32	HIF WLAN Firmware TX Event 0 Status Register
38130120	HWFTE0ER	32	HIF WLAN Firmware TX Event 0 Enable Register
38130130	HWFRE0SR	32	HIF WLAN Firmware RX Event 0 Status Register
38130134	HWFRE1SR	32	HIF WLAN Firmware RX Event 1 Status Register
38130140	HWFRE0ER	32	HIF WLAN Firmware RX Event 0 Enable Register
38130144	HWFRE1ER	32	HIF WLAN Firmware RX Event 1 Enable Register
38130150	HWFICR	32	HIF WLAN Firmware Interrupt Control Register
38130154	HWFCR	32	HIF WLAN Firmware Control Register
38130158	HWTDCR	32	HIF WLAN TX DMA Control Register
3813015C	HWTPCCR	32	HIF WLAN TX Packet Count Control Register
38130164	HWFTQ1SAR	32	HIF WLAN Firmware TX Queue 1 Start Address Register
38130180	HWFRQ0SAR	32	HIF WLAN Firmware RX Queue 0 Start Address Register
38130184	HWFRQ1SAR	32	HIF WLAN Firmware RX Queue 1 Start Address Register
381301A0	H2DRM0R	32	Host to Device Receive Mailbox 0 Register
381301A4	H2DRM1R	32	Host to Device Receive Mailbox 1 Register
381301A8	D2HSM0R	32	Device to Host Send Mailbox 0 Register
381301AC	D2HSM1R	32	Device to Host Send Mailbox 1 Register
381301C0	HWRQ0CR	32	HIF WLAN RX Queue 0 Control Register
381301C4	HWRQ1CR	32	HIF WLAN RX Queue 1 Control Register
381301EC	HWFIOCDR	32	HIF WLAN Firmware GPD IOC bit Disable Register

38130000 HGFCR HIF Global Firmware Configuration Register 40020041

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name		SW_SEL_CLKBLC	SW_SET_CLK_NONBLC	PAD_CR_SET_BY_FW	PB_HCLK_DIS	EHPI_HCLK_DIS	SPI_HCLK_DIS	SDIO_HCLK_DIS						FORCE_SD_HS	HCLK_NO_GATED	INT_TER_CYC_MASK
Type		RW	RW	RW	RW	RW	RW	RW						RW	RW	RW
Reset		1	0	0	0	0	0	0						0	1	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name						SDCTL_BUSY	CARD_IS_18V	HINT_AS_FW_OB		SDIO_PIO_SEL	EHPI_MODE	SPI_MODE		DB_HIF_SEL		
Type						RO	RW	RW		RO	RO	RO		RO		
Reset						0	0	0		1	0	0		0	0	1

Bit(s)	Name	Description
30	SW_SEL_CLKBLC	**SW enables this bit to take over the control of balance and non-balance sd clock for pad macro. The need for non-balance sd clock for pad macro is due to tight output timing specification. The default setting is controlled by the hardware to decide the clock balance. However, we keep the flexibility for SW to decide the balance or nonbalance clock tree for the SDIO pad macro.** 0: The balance/non-balance sd clock for pad macro is controlled by HW 1: The balance/non-balance sd clock for pad macro is controlled by SW
29	SW_SET_CLK_NONBLC	**SW enables this bit to use balance and non-balance sd clock for pad macro.** 0: Use balance sd clock for pad macro 1: Use non-balance sd clock for pad macro
28	PAD_CR_SET_BY_FW	**Enable the pad macro control register test mode. Firmware writes this bit and then firmware gets the ownership to access these pad macro control registers.** 0: Pad macro control register set by host driver (normal mode) 1: Pad macro control register set by firmware (test mode)
27	PB_HCLK_DIS	**It is used to disable the AHB clock for PIO-Based funciton design. It would be set when PIO-Based function is not used in some specific configuration. Otherwise, PIO-Based function cannot work normally.**

Bit(s)	Name	Description
		0: AHB clock is not disabled 1: Disable AHB clock
26	EHPI_HCLK_DIS	**It is used to disable the AHB clock for EHPI interface. It would be set when EHPI is not used in some specific configuration. Otherwise, EHPI cannot work normally.** 0: AHB clock is not disabled 1: Disable AHB clock
25	SPI_HCLK_DIS	**It is used to disable the AHB clock for SPI interface. It would be set when SPI is not used in some specific configuration. Otherwise, SPI cannot work normally.** 0: AHB clock is not disabled 1: Disable AHB clock
24	SDIO_HCLK_DIS	**It is used to disable the AHB clock for SDIO1 interface. It would be set when SDIO is not used in some specific configuration. Otherwise, SDIO cannot work normally.** 0: AHB clock is not disabled 1: Disable AHB clock
18	FORCE_SD_HS	**Note that we also provide card capability setting method to force high speed, you may use external effuse or r/w interface to enable this function according to IP configuration.** 0: SDIO is in the operation mode specified in EHS of CCCR 1: FORCE SDIO to operate in high speed despite the value of EHS of CCCR
17	HCLK_NO_GATED	0: SDIO controller would gate some part of AHB clock inside automatically for the unused period 1: AHB clock inside SDIO controller would always turn-on
16	INT_TER_CYC_MASK	**This field is used to determine should SDIO drive high to bus bit1 during the interrupt termination cycle.** 0 : always drive high during the termination cycle 1: drive high during the termination cycle only it is in interrupt period.
10	SDCTL_BUSY	**Indicate SDIO controller busy or not.** 0: SDIO controller is not busy 1: SDIO controller is still busy
9	CARD_IS_18V	**Firmware write 1 to this field to show the voltage switch process is done and the card is in 1.8v state.** 0: card is not in 1.8v state 1: card is in 1.8v state (UHS mode)

Bit(s)	Name	Description
8	HINT_AS_FW_OB	**Use interrupt to host as a firmware own back control** 0: Host interrupt to host would NOT trigger firmware own back 1: Host interrupt to host would trigger firmware own back
6	SDIO_PIO_SEL	**Host interface for SDIO PIO-Based function in used** 0: SDIO PIO mode is not used 1: SDIO PIO mode is used
5	EHPI_MODE	**This bit indicates that if EHPI8-mode or EHPI16-mode is used.** 0: EHPI16-mode of EHPI is used 1: EHPI8-mode of EHPI is used.
4	SPI_MODE	**This bit indicates that if TI-mode or Motor-mode is used.** 0: Motor-mode of SPI is used 1: TI-mode of SPI is used.
2:0	DB_HIF_SEL	**Host interface for DMA-Based function in used;** 0x1: SDIO1 0x2: SPI 0x4: EHPI

38130004 HGFISR HIF Global Firmware Interrupt Status Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					SD1_SET_DS_INT	SD1_SET_XTAL_UPD_INT	CHG_TO_18V_REQ_INT	CRC_ERROR_INT	PB_INT	DB_INT	SDIO_SET_ABT	SDIO_SET_RES	DRV_SET_PB_IOE	DRV_SET_DB_IOE	DRV_CLR_PB_IOE	DRV_CLR_DB_IOE
Type					W1C	W1C	W1C	W1C	RO	RO	W1C	W1C	W1C	W1C	W1C	W1C
Reset					0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
11	SD1_SET_DS_INT	**This bit is for SDIO interface only.** If host set the CCCR deep sleep register, this bit would be set. After firmware detected this event, it can know the information that host want device to enter deep sleep mode. Firmware can clear this bit by write 1. Write 0 is meaningless.
10	SD1_SET_XTAL_UPD_INT	**This bit is for SDIO interface only.** If host set the CCCR xtal frequency update register, this bit would be set. After firmware detected this event, it can know the information that host has updated the xtal frequency. Firmware can then read HGH2DR to know the updated frequency value. Firmware can clear this bit by write 1. Write 0 is meaningless.
9	CHG_TO_18V_REQ_INT	**Host send command 11 to request device to change voltage to 1.8v. Firmware receives the interrupt or polling the status to start the voltage switch. After the voltage switch is done, firmware would write the card is in 1.8v status to HGFCR**
8	CRC_ERROR_INT	**The status bit of TX data port CRC error interrupt.**
7	PB_INT	**The status bit of PIO-Based function firmware interrupt.**
6	DB_INT	**The status bit of DMA-Based function firmware interrupt.**
5	SDIO_SET_ABT	**SDIO write 1 to SDIO CCCR.ABORT to abort transaction. After firmware detected this event, the TX and RX queue should be stop by firmware and the data in buffer should be discarded.** Firmware can clear this bit by write 1. Write 0 is meaningless.
4	SDIO_SET_RES	**SDIO write 1 to SDIO CCCR.RES to assert software reset. After firmware detected this event, it should disable the sub-systems on SDIO interface.** Firmware can clear this bit by write 1. Write 0 is meaningless.
3	DRV_SET_PB_IOE	**This bit is for SDIO interface only.** If host set the CCCR.IOE bit of PIO-Based functional block, this bit will be set. After firmware detected this event, it should enable sub-system which uses PIO-Based function. Firmware can clear this bit by write 1. Write 0 is meaningless.
2	DRV_SET_DB_IOE	**This bit is for SDIO interface only.** If host set the CCCR.IOE bit of DMA-Based functional block, this bit will be set. After firmware detected this event, it should enable sub-system which uses DMA-Based function. Firmware can clear this bit by write 1. Write 0 is meaningless.
1	DRV_CLR_PB_IOE	**This bit is for SDIO interface only.** If host clear the CCCR.IOE bit of PIO-Based functional block, this bit will be set. After firmware detected this event, it should disable sub-system which uses PIO-Based function.

Bit(s)	Name	Description
		Firmware can clear this bit by write 1. Write 0 is meaningless.
0	DRV_CLR_DB_IOE	**This bit is for SDIO interface only.** If host clear the CCCR.IOE bit of DMA-Based functional block, this bit will be set. After firmware detected this event, it should disable sub-system which uses DMA-Based function. Firmware can clear this bit by write 1. Write 0 is meaningless.

38130008 HGFIER HIF Global Firmware Interrupt Enable Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					SD1_SET_DS_INT_EN	SD1_SET_XTAL_UPD_INT_EN	CHG_TO_18V_REQ_INT_EN	CRC_ERROR_INT_EN	PB_INT_EN	DB_INT_EN	SDIO_SET_ABT_INT_EN	SDIO_SET_RES_INT_EN	DRV_SET_PB_IOE_INT_EN	DRV_SET_DB_IOE_INT_EN	DRV_CLR_PB_IOE_INT_EN	DRV_CLR_DB_IOE_INT_EN
Type					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset					0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
11	SD1_SET_DS_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
10	SD1_SET_XTAL_UPD_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
9	CHG_TO_18V_REQ_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_

Bit(s)	Name	Description
		0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
8	CRC_ERROR_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
7	PB_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
6	DB_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
5	SDIO_SET_ABT_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
4	SDIO_SET_RES_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
3	DRV_SET_PB_IOE_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
2	DRV_SET_DB_IOE_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_

Bit(s)	Name	Description
		0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
1	DRV_CLR_PB_IOE_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
0	DRV_CLR_DB_IOE_INT_EN	**Common firmware interrupt output control for each bit corresponding to bits defined in HGFISR._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_

38130100 HWFISR HIF WLAN Firmware Interrupt Status Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	H2D_SW_INT															
Type	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name			RX_EVENT_1	RX_EVENT_0				TX_EVENT_0				WR_TIMEOUT_INT	RD_TIMEOUT_INT	D2HSM2R_RD_INT	DRV_CLR_FW_OWN	DRV_SET_FW_OWN
Type			RO	RO				RO				W1C	W1C	W1C	W1C	W1C
Reset			0	0				0				0	0	0	0	0

Bit(s)	Name	Description
31:16	H2D_SW_INT	**This field is used for software interrupt for WLAN function._x000D_** Host driver write 1s to WSICR [31:16] will set corresponding bit field._x000D_
13	RX_EVENT_1	**If there is any interrupt asserted in HWFRE1SR, this bit will be asserted. The bit will be de-asserted after software driver clears the interrupt event in HWFRE1SR.**
12	RX_EVENT_0	**If there is any interrupt asserted in HWFRE0SR, this bit will be asserted. The bit will be de-asserted after software driver clears the interrupt event in HWFRE0SR.**

Bit(s)	Name	Description
8	TX_EVENT_0	**If there is any interrupt asserted in HWFTE0SR, this bit will be asserted. The bit will be de-asserted after software driver clears the interrupt event in HWFTE0SR.**
4	WR_TIMEOUT_INT	**If host write data and device cannot receive the data well in pre-defined period, the write timeout interrupt will be triggered. Firmware should receive the write timeout interrupt and tx_overflow interrupt simultaneously.**
3	RD_TIMEOUT_INT	**If host read data and device cannot prepare the data well in pre-defined period, the timeout interrupt will be triggered. Firmware should receive the timeout interrupt and rx_underflow interrupt simultaneously.**
2	D2HSM2R_RD_INT	**This interrupt would be set when host read the D2HRM2R register. Note that for pre-read of EHPI function, this interrupt would be set if host write the D2HRM2R register despite D2HRM2R is a read-only register. _x000D_**
1	DRV_CLR_FW_OWN	**If software driver write 1 to "WHLPCR.FW_OWN_REQ_CLR", this bit will be set to 1, and it means software driver want firmware to return the ownership of WLAN sub-system. Firmware must wake-up WLAN sub-system from sleep mode and write 1 to HWFICR.FW_OWN_BACK_INT_SET._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
0	DRV_SET_FW_OWN	**If software driver write 1 to "WHLPCR.FW_OWN_REQ_SET", this bit will be set to 1, and it means software driver transfer the ownership of WLAN sub-system to firmware. Firmware can force WLAN sub-system into sleep mode._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_

38130104 HWFIER HIF WLAN Firmware Interrupt Enable Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	H2D_SW_INT_EN															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name			RX_EVENT_1_INT_EN	RX_EVENT_0_INT_EN				TX_EVENT_0_INT_EN				WR_TIMEOUT_INT_EN	RD_TIMEOUT_INT_EN	D2HSM2R_RD_INT_EN	DRV_CLR_FW_OWN_INT_EN	DRV_SET_FW_OWN_INT_EN
Type			RW	RW				RW				RW	RW	RW	RW	RW
Reset			0	0				0				0	0	0	0	0

Bit(s)	Name	Description
31:16	H2D_SW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
13	RX_EVENT_1_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
12	RX_EVENT_0_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
8	TX_EVENT_0_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
4	WR_TIMEOUT_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
3	RD_TIMEOUT_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
2	D2HSM2R_RD_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
1	DRV_CLR_FW_OWN_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_

Bit(s)	Name	Description
0	DRV_SET_FW_OWN_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_

38130110 HWFTE0SR HIF WLAN Firmware TX Event 0 Status Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TX7D_LEN_ERR	TX6D_LEN_ERR	TX5D_LEN_ERR	TX4D_LEN_ERR	TX3D_LEN_ERR	TX2D_LEN_ERR	TX1D_LEN_ERR	TX0D_LEN_ERR	TX7D_CHKSUM_ERR	TX6D_CHKSUM_ERR	TX5D_CHKSUM_ERR	TX4D_CHKSUM_ERR	TX3D_CHKSUM_ERR	TX2D_CHKSUM_ERR	TX1D_CHKSUM_ERR	TX0D_CHKSUM_ERR
Type	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							TX1_OVERFLOW	TX0_OVERFLOW	TX7_RDY	TX6_RDY	TX5_RDY	TX4_RDY	TX3_RDY	TX2_RDY	TX1_RDY	TX0_RDY
Type							W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset							0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31	TX7D_LEN_ERR	**TX queue 7 descriptor length error** When host tx data number is more than total allow buffer length, the length error interrupt will be triggered.
30	TX6D_LEN_ERR	**TX queue 6 descriptor length error** When host tx data number is more than total allow buffer length, the length error interrupt will be triggered.
29	TX5D_LEN_ERR	**TX queue 5 descriptor length error** When host tx data number is more than total allow buffer length, the length error interrupt will be triggered.
28	TX4D_LEN_ERR	**TX queue 4 descriptor length error** When host tx data number is more than total allow buffer length, the length error interrupt will be triggered.
27	TX3D_LEN_ERR	**TX queue 3 descriptor length error** When host tx data number is more than total allow buffer length, the length error interrupt will be triggered.

Bit(s)	Name	Description
26	TX2D_LEN_ERR	**TX queue 2 descriptor length error** When host tx data number is more than total allow buffer length, the length error interrupt will be triggered.
25	TX1D_LEN_ERR	**TX queue 1 descriptor length error** When host tx data number is more than total allow buffer length, the length error interrupt will be triggered.
24	TX0D_LEN_ERR	**TX queue 0 descriptor length error** When host tx data number is more than total allow buffer length, the length error interrupt will be triggered.
23	TX7D_CHKSUM_ERR	**TX queue 7 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each TX descriptor before data movement. This interrupt will be generated if TX descriptor checksum error._x000D_
22	TX6D_CHKSUM_ERR	**TX queue 6 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each TX descriptor before data movement. This interrupt will be generated if TX descriptor checksum error._x000D_
21	TX5D_CHKSUM_ERR	**TX queue 5 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each TX descriptor before data movement. This interrupt will be generated if TX descriptor checksum error._x000D_
20	TX4D_CHKSUM_ERR	**TX queue 4 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each TX descriptor before data movement. This interrupt will be generated if TX descriptor checksum error._x000D_
19	TX3D_CHKSUM_ERR	**TX queue 3 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each TX descriptor before data movement. This interrupt will be generated if TX descriptor checksum error._x000D_
18	TX2D_CHKSUM_ERR	**TX queue 2 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each TX descriptor before data movement. This interrupt will be generated if TX descriptor checksum error._x000D_
17	TX1D_CHKSUM_ERR	**TX queue 1 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each TX descriptor before data movement. This interrupt will be generated if TX descriptor checksum error._x000D_
16	TX0D_CHKSUM_ERR	**TX queue 0 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each TX descriptor before data movement. This interrupt will be generated if TX descriptor checksum error._x000D_

Bit(s)	Name	Description
9	TX1_OVERFLOW	**Data overflow of WLAN TX1 port._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
8	TX0_OVERFLOW	**Data overflow of WLAN TX0 port._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
7	TX7_RDY	**If a complete frame has been moved to WLAN TX7 queue form host and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
6	TX6_RDY	**If a complete frame has been moved to WLAN TX6 queue form host and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
5	TX5_RDY	**If a complete frame has been moved to WLAN TX5 queue form host and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
4	TX4_RDY	**If a complete frame has been moved to WLAN TX4 queue form host and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
3	TX3_RDY	**If a complete frame has been moved to WLAN TX3 queue form host and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
2	TX2_RDY	**If a complete frame has been moved to WLAN TX2 queue form host and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
1	TX1_RDY	**If a complete frame has been moved to WLAN TX1 queue form host and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
0	TX0_RDY	**If a complete frame has been moved to WLAN TX0 queue form host and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_

38130120 HWFTE0ER HIF WLAN Firmware TX Event 0 Enable Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TX7D_LEN_ERR_INT_EN	TX6D_LEN_ERR_INT_EN	TX5D_LEN_ERR_INT_EN	TX4D_LEN_ERR_INT_EN	TX3D_LEN_ERR_INT_EN	TX2D_LEN_ERR_INT_EN	TX1D_LEN_ERR_INT_EN	TX0D_LEN_ERR_INT_EN	TX7D_CHKSUM_ERR_INT_EN	TX6D_CHKSUM_ERR_INT_EN	TX5D_CHKSUM_ERR_INT_EN	TX4D_CHKSUM_ERR_INT_EN	TX3D_CHKSUM_ERR_INT_EN	TX2D_CHKSUM_ERR_INT_EN	TX1D_CHKSUM_ERR_INT_EN	TX0D_CHKSUM_ERR_INT_EN
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							TX1_OVERFLOW_INT_EN	TX0_OVERFLOW_INT_EN	TX7_RDY_INT_EN	TX6_RDY_INT_EN	TX5_RDY_INT_EN	TX4_RDY_INT_EN	TX3_RDY_INT_EN	TX2_RDY_INT_EN	TX1_RDY_INT_EN	TX0_RDY_INT_EN
Type							RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset							0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31	TX7D_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
30	TX6D_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
29	TX5D_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
28	TX4D_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
27	TX3D_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_

Bit(s)	Name	Description
		0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
26	TX2D_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
25	TX1D_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
24	TX0D_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
23	TX7D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
22	TX6D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
21	TX5D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
20	TX4D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
19	TX3D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_

Bit(s)	Name	Description
		0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
18	TX2D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
17	TX1D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
16	TX0D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
9	TX1_OVERFLOW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
8	TX0_OVERFLOW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
7	TX7_RDY_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
6	TX6_RDY_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
5	TX5_RDY_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_

Bit(s)	Name	Description
		0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
4	TX4_RDY_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
3	TX3_RDY_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
2	TX2_RDY_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
1	TX1_RDY_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
0	TX0_RDY_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_

38130130 HWFRE0SR HIF WLAN Firmware RX Event 0 Status Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name					RX_LEN_FIFO3_OVERFLOW	RX_LEN_FIFO2_OVERFLOW	RX_LEN_FIFO1_OVERFLOW	RX_LEN_FIFO0_OVERFLOW					RX3D_CHKSUM_ERR	RX2D_CHKSUM_ERR	RX1D_CHKSUM_ERR	RX0D_CHKSUM_ERR
Type					W1C	W1C	W1C	W1C					W1C	W1C	W1C	W1C
Reset					0	0	0	0					0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Name					RX3_UNDERFLOW	RX2_UNDERFLOW	RX1_UNDERFLOW	RX0_UNDERFLOW					RX3_DONE	RX2_DONE	RX1_DONE	RX0_DONE
Type					W1C	W1C	W1C	W1C					W1C	W1C	W1C	W1C
Reset					0	0	0	0					0	0	0	0

Bit(s)	Name	Description
27	RX_LEN_FIFO3_OVERFLOW	**RX length FIFO 3 over-flow_x000D_** This interrupt will be generated whenever FW attempt to set RX length FIFO by HWRQ3CR when the packet length FIFO is already full leading to FIFO overflow._x000D_ The entry in packet length FIFO will be push-in by FW, and be pop-out when corresponding RX length is read by host driver._x000D_
26	RX_LEN_FIFO2_OVERFLOW	**RX length FIFO 2 over-flow_x000D_** This interrupt will be generated whenever FW attempt to set RX length FIFO by HWRQ3CR when the packet length FIFO is already full leading to FIFO overflow._x000D_ The entry in packet length FIFO will be push-in by FW, and be pop-out when corresponding RX length is read by host driver._x000D_
25	RX_LEN_FIFO1_OVERFLOW	**RX length FIFO 1 over-flow_x000D_** This interrupt will be generated whenever FW attempt to set RX length FIFO by HWRQ3CR when the packet length FIFO is already full leading to FIFO overflow._x000D_ The entry in packet length FIFO will be push-in by FW, and be pop-out when corresponding RX length is read by host driver._x000D_
24	RX_LEN_FIFO0_OVERFLOW	**RX length FIFO 0 over-flow_x000D_** This interrupt will be generated whenever FW attempt to set RX length FIFO by HWRQ3CR when the packet length FIFO is already full leading to FIFO overflow._x000D_ The entry in packet length FIFO will be push-in by FW, and be pop-out when corresponding RX length is read by host driver._x000D_
19	RX3D_CHKSUM_ERR	**RX 3 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each RX descriptor before data movement. This interrupt will be generated if RX descriptor checksum error._x000D_
18	RX2D_CHKSUM_ERR	**RX 2 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each RX descriptor before data movement. This interrupt will be generated if RX descriptor checksum error._x000D_
17	RX1D_CHKSUM_ERR	**RX 1 descriptor checksum error_x000D_**

Bit(s)	Name	Description
		When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each RX descriptor before data movement. This interrupt will be generated if RX descriptor checksum error._x000D_
16	RX0D_CHKSUM_ERR	**RX 0 descriptor checksum error_x000D_** When HWFCR. TRX_DESC_CHKSUM_EN is enabled; HW will validate the checksum value of each RX descriptor before data movement. This interrupt will be generated if RX descriptor checksum error._x000D_
11	RX3_UNDERFLOW	**Data underflow of WLAN RX3 port._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
10	RX2_UNDERFLOW	**Data underflow of WLAN RX2 port._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
9	RX1_UNDERFLOW	**Data underflow of WLAN RX1 port._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
8	RX0_UNDERFLOW	**Data underflow of WLAN RX0 port._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
3	RX3_DONE	**If a complete frame has been moved to host from WLAN RX3 queue (which also implies the corresponding entry is pop-out from RX3 length FIFO), this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
2	RX2_DONE	**If a complete frame has been moved to host from WLAN RX2 queue (which also implies the corresponding entry is pop-out from RX2 length FIFO), this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
1	RX1_DONE	**If a complete frame has been moved to host from WLAN RX1 queue (which also implies the corresponding entry is pop-out from RX0 length FIFO), this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
0	RX0_DONE	**If a complete frame has been moved to host from WLAN RX0 queue (which also implies the corresponding entry is pop-out from RX0 length FIFO), this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_

38130134 HWFRE1SR HIF WLAN Firmware RX Event 1 Status Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																

Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					RX3_LEN_ERR	RX2_LEN_ERR	RX1_LEN_ERR	RX0_LEN_ERR					RX3_OWN_CLEAR_DONE	RX2_OWN_CLEAR_DONE	RX1_OWN_CLEAR_DONE	RX0_OWN_CLEAR_DONE
Type					W1C	W1C	W1C	W1C					W1C	W1C	W1C	W1C
Reset					0	0	0	0					0	0	0	0

Bit(s)	Name	Description
11	RX3_LEN_ERR	**RX queue 3 descriptor length error** If RX3 both extension length and rx data length are zero in GPD or BD, the error interrupt will be triggered. Firmware can clear this bit by write 1. Write 0 is meaningless.
10	RX2_LEN_ERR	**RX queue 2 descriptor length error** If RX2 both extension length and rx data length are zero in GPD or BD, the error interrupt will be triggered. Firmware can clear this bit by write 1. Write 0 is meaningless.
9	RX1_LEN_ERR	**RX queue 1 descriptor length error** If RX1 both extension length and rx data length are zero in GPD or BD, the error interrupt will be triggered. Firmware can clear this bit by write 1. Write 0 is meaningless.
8	RX0_LEN_ERR	**RX queue 0 descriptor length error** If RX0 both extension length and rx data length are zero in GPD or BD, the error interrupt will be triggered. Firmware can clear this bit by write 1. Write 0 is meaningless.
3	RX3_OWN_CLEAR_DONE	**If a complete frame has been moved to internal FIFO from WLAN RX3 queue and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
2	RX2_OWN_CLEAR_DONE	**If a complete frame has been moved to internal FIFO from WLAN RX2 queue and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_
1	RX1_OWN_CLEAR_DONE	**If a complete frame has been moved to internal FIFO from WLAN RX1 queue and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_

Bit(s)	Name	Description
0	RX0_OWN_CLEAR_DONE	**If a complete frame has been moved to internal FIFO from WLAN RX0 queue and the ownership bit of the buffer descriptor is cleared, this bit will be set._x000D_** Firmware can clear this bit by write 1. Write 0 is meaningless._x000D_

38130140 HWFRE0ER HIF WLAN Firmware RX Event 0 Enable Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name					RX_LEN_FIFO3_OVERFLOW_INT_EN	RX_LEN_FIFO2_OVERFLOW_INT_EN	RX_LEN_FIFO1_OVERFLOW_INT_EN	RX_LEN_FIFO0_OVERFLOW_INT_EN					RX3D_CHKSUM_ERR_INT_EN	RX2D_CHKSUM_ERR_INT_EN	RX1D_CHKSUM_ERR_INT_EN	RX0D_CHKSUM_ERR_INT_EN
Type					RW	RW	RW	RW					RW	RW	RW	RW
Reset					0	0	0	0					0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					RX3_UNDERFLOW_INT_EN	RX2_UNDERFLOW_INT_EN	RX1_UNDERFLOW_INT_EN	RX0_UNDERFLOW_INT_EN					RX3_DONE_INT_EN	RX2_DONE_INT_EN	RX1_DONE_INT_EN	RX0_DONE_INT_EN
Type					RW	RW	RW	RW					RW	RW	RW	RW
Reset					0	0	0	0					0	0	0	0

Bit(s)	Name	Description
27	RX_LEN_FIFO3_OVERFLOW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
26	RX_LEN_FIFO2_OVERFLOW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
25	RX_LEN_FIFO1_OVERFLOW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
24	RX_LEN_FIFO0_OVERFLOW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_**

Bit(s)	Name	Description
		If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
19	RX3D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
18	RX2D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
17	RX1D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
16	RX0D_CHKSUM_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
11	RX3_UNDERFLOW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
10	RX2_UNDERFLOW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
9	RX1_UNDERFLOW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_

Bit(s)	Name	Description
8	RX0_UNDERFLOW_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
3	RX3_DONE_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
2	RX2_DONE_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
1	RX1_DONE_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
0	RX0_DONE_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_

38130144 HWFRE1ER HIF WLAN Firmware RX Event 1 Enable Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					RX3_LEN_ERR_INT_EN	RX2_LEN_ERR_INT_EN	RX1_LEN_ERR_INT_EN	RX0_LEN_ERR_INT_EN					RX3_OWN_CLEAR_DONE_INT_EN	RX2_OWN_CLEAR_DONE_INT_EN	RX1_OWN_CLEAR_DONE_INT_EN	RX0_OWN_CLEAR_DONE_INT_EN

Type					RW	RW	RW	RW					RW	RW	RW	RW
Reset					0	0	0	0					0	0	0	0

Bit(s)	Name	Description
11	RX3_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
10	RX2_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
9	RX1_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
8	RX0_LEN_ERR_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
3	RX3_OWN_CLEAR_DONE_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
2	RX2_OWN_CLEAR_DONE_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_
1	RX1_OWN_CLEAR_DONE_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_

Bit(s)	Name	Description
0	RX0_OWN_CLEAR_DONE_INT_EN	**WLAN firmware interrupt output control for each bit._x000D_** If the related bit is_x000D_ 0: Disable the related bit interrupt output. 1: Enable the related bit interrupt output._x000D_

38130150 HWFICR HIF WLAN Firmware Interrupt Control Register 00000010

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	D2H_SW_INT_SET															
Type	W1S															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	D2H_SW_INT_SET											FW_OWN_BACK_INT_SET				
Type	W1S											W1S				
Reset	0	0	0	0	0	0	0	0				1				

Bit(s)	Name	Description
31:8	D2H_SW_INT_SET	**Firmware writes 1s will set WHISR.D2H_SW_INT. Write 0 is meaningless.** Read always return 0. This is used as a communication between FW to driver, with interrupt functionality to host driver HIF.
4	FW_OWN_BACK_INT_SET	**Firmware writes 1 will set WHISR.FW_OWN_BACK_INT. Write 0 is meaningless. It will also clear WLAN_FW_OWN bit and set WHLPCR.WLAN_DRV_OWN bit._x000D_** If driver requests firmware to return the ownership or firmware wants to wakeup driver, firmware can set this bit._x000D_ Read will get the status of WLAN_FW_OWN bit._x000D_ _x000D_ WLAN_FW_OWN indicates that WLAN firmware has the ownership of chip WLAN sub-system._x000D_ This bit will be cleared by firmware written 1 to HWFICR.FW_OWN_BACK_INT_SET or any WLAN driver-domain interrupt._x000D_ 0: WLAN firmware doesn't have ownership

Bit(s)	Name	Description
		1: WLAN firmware has ownership_x000D_

38130154 HWFCR HIF WLAN Firmware Control Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							RX_NO_TAIL	TX_NO_HEADER	RX_UDP_CS_OFLD_EN	RX_TCP_CS_OFLD_EN	RX_IPV4_CS_OFLD_EN	RX_IPV6_CS_OFLD_EN	TX_CS_OFLD_EN	TRX_DESC_CHKSUM_12B	TRX_DESC_CHKSUM_EN	W_FUNC_RDY
Type							RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset							0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
9	RX_NO_TAIL	**RX packet tail is used for sending checksum offload status. If the checksum offload hardware is not configured, the tail would be 4B zero. Firmware can write 1 to make the RX packet tail not sent to host.** 0: RX packet tail would be sent to host 1: RX packet tail would not be sent to host
8	TX_NO_HEADER	**Firmware write 1 to this filed to make the TX packet header sent by host not written to AHB bus.** 0: TX packet header from host would be written to AHB bus 1: TX packet header from host would not be written to AHB bus
7	RX_UDP_CS_OFLD_EN	**Enable RX UDP checksum verification function_x000D_** When enabled, packets checksum of RX packet with UDP header will be calculated, and verified with the field in original RX packet. The verified status will be padding in the last DWORD of the RX packet._x000D_
6	RX_TCP_CS_OFLD_EN	**Enable RX TCP checksum verification function_x000D_** When enabled, packets checksum of RX packet with TCP header will be calculated, and verified with the field in original RX packet. The verified status will be padding in the last DWORD of the RX packet._x000D_
5	RX_IPV4_CS_OFLD_EN	**Enable RX IPv4 checksum verification function_x000D_**

Bit(s)	Name	Description
		When enabled, packets checksum of RX packet with IPv4 header will be calculated, and verified with the field in original RX packet. The verified status will be padding in the last DWORD of the RX packet._x000D_
4	RX_IPV6_CS_OFLD_EN	**Enable RX IPv6 checksum (without extension header) verification function_x000D_** When enabled, packets checksum of RX packet with IPv6 header will be calculated, and verified with the field in original RX packet. The verified status will be padding in the last DWORD of the RX packet._x000D_
3	TX_CS_OFLD_EN	**Enable TX IPV6/IPV4/TCP/UDP checksum generation function_x000D_**
2	TRX_DESC_CHKSUM_12B	**Firmware write 1 to this filed to change the descriptor checksum calculation method to 12B. The default calculation method is based on the 16B descriptor checksum.** 0: Descriptor checksum calculation is based on first 16B. 1: Descriptor checksum calculation is based on first 12B
1	TRX_DESC_CHKSUM_EN	**Enable TX/ RX descriptor checksum for debug purpose._x000D_** HW will validate if the summation of descriptor checksum is 0xff before data movement for the descriptor. If it is invalid, corresponding interrupt status (TXD_CHKSUM_ERR/ RXD_CHKSUM_ERR) will be generated._x000D_
0	W_FUNC_RDY	**Indicate the WLAN functional block's current status. If WLAN functional block has finished its initial procedure and it is ready for normal operation, firmware should set this bit. If WLAN functional block was disabled, this bit should be cleared._x000D_** This is a sticky bit of WCIR.W_FUNC_RDY._x000D_ 0: WLAN functional block is not ready for normal operation. 1: WLAN functional block is ready for normal operation._x000D_

38130158 HWTDCR HIF WLAN TX DMA Control Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TXQ7_DMA_STATUS	TXQ6_DMA_STATUS	TXQ5_DMA_STATUS	TXQ4_DMA_STATUS	TXQ3_DMA_STATUS	TXQ2_DMA_STATUS	TXQ1_DMA_STATUS	TXQ0_DMA_STATUS	TXQ7_DMA_RUM	TXQ6_DMA_RUM	TXQ5_DMA_RUM	TXQ4_DMA_RUM	TXQ3_DMA_RUM	TXQ2_DMA_RUM	TXQ1_DMA_RUM	TXQ0_DMA_RUM
Type	RO	RO	RO	RO	RO	RO	RO	RO	W1S	W1S	W1S	W1S	W1S	W1S	W1S	W1S
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Name	TXQ7_DMA_START	TXQ6_DMA_START	TXQ5_DMA_START	TXQ4_DMA_START	TXQ3_DMA_START	TXQ2_DMA_START	TXQ1_DMA_START	TXQ0_DMA_START	TXQ7_DMA_STOP	TXQ6_DMA_STOP	TXQ5_DMA_STOP	TXQ4_DMA_STOP	TXQ3_DMA_STOP	TXQ2_DMA_STOP	TXQ1_DMA_STOP	TXQ0_DMA_STOP
Type	W1S	W1S	W1S	W1S	W1S	W1S	W1S	W1S	W1S	W1S	W1S	W1S	W1S	W1S	W1S	W1S
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31	TXQ7_DMA_STATUS	**Read for the TX4 queue DMA status._x000D_** When the HIF Controller is reset, the queue is in the inactive state by default. After receiving a START or RESUME command and executing it without error, the queue enters the active state. When the queue is empty or stopped by a STOP command, it returns to the inactive state._x000D_ 0: inactive 1: active_x000D_
30	TXQ6_DMA_STATUS	**Read for the TX6 queue DMA status._x000D_** When the HIF Controller is reset, the queue is in the inactive state by default. After receiving a START or RESUME command and executing it without error, the queue enters the active state. When the queue is empty or stopped by a STOP command, it returns to the inactive state._x000D_ 0: inactive 1: active_x000D_
29	TXQ5_DMA_STATUS	**Read for the TX5 queue DMA status._x000D_** When the HIF Controller is reset, the queue is in the inactive state by default. After receiving a START or RESUME command and executing it without error, the queue enters the active state. When the queue is empty or stopped by a STOP command, it returns to the inactive state._x000D_ 0: inactive 1: active_x000D_
28	TXQ4_DMA_STATUS	**Read for the TX4 queue DMA status._x000D_** When the HIF Controller is reset, the queue is in the inactive state by default. After receiving a START or RESUME command and executing it without error, the queue enters the active state. When the queue is empty or stopped by a STOP command, it returns to the inactive state._x000D_ 0: inactive 1: active_x000D_
27	TXQ3_DMA_STATUS	**Read for the TX3 queue DMA status._x000D_**

Bit(s)	Name	Description
		When the HIF Controller is reset, the queue is in the inactive state by default. After receiving a START or RESUME command and executing it without error, the queue enters the active state. When the queue is empty or stopped by a STOP command, it returns to the inactive state._x000D_ 0: inactive 1: active_x000D_
26	TXQ2_DMA_STATUS	**Read for the TX2 queue DMA status._x000D_** When the HIF Controller is reset, the queue is in the inactive state by default. After receiving a START or RESUME command and executing it without error, the queue enters the active state. When the queue is empty or stopped by a STOP command, it returns to the inactive state._x000D_ 0: inactive 1: active_x000D_
25	TXQ1_DMA_STATUS	**Read for the TX1 queue DMA status._x000D_** When the HIF Controller is reset, the queue is in the inactive state by default. After receiving a START or RESUME command and executing it without error, the queue enters the active state. When the queue is empty or stopped by a STOP command, it returns to the inactive state._x000D_ 0: inactive 1: active_x000D_
24	TXQ0_DMA_STATUS	**Read for the TX0 queue DMA status._x000D_** When the HIF Controller is reset, the queue is in the inactive state by default. After receiving a START or RESUME command and executing it without error, the queue enters the active state. When the queue is empty or stopped by a STOP command, it returns to the inactive state._x000D_ 0: inactive 1: active_x000D_
23	TXQ7_DMA_RUM	**Resume the TX7 queue DMA to operate._x000D_** The DMA will reload the chain descriptor from the current address._x000D_ Firmware writes 1 to enable the DMA. Write 0 is meaningless._x000D_ Read always return 0._x000D_
22	TXQ6_DMA_RUM	**Resume the TX6 queue DMA to operate._x000D_** The DMA will reload the chain descriptor from the current address._x000D_ Firmware writes 1 to enable the DMA. Write 0 is meaningless._x000D_

Bit(s)	Name	Description
		Read always return 0._x000D_
21	TXQ5_DMA_RUM	**Resume the TX5 queue DMA to operate._x000D_** The DMA will reload the chain descriptor from the current address._x000D_ Firmware writes 1 to enable the DMA. Write 0 is meaningless._x000D_ Read always return 0._x000D_
20	TXQ4_DMA_RUM	**Resume the TX4 queue DMA to operate._x000D_** The DMA will reload the chain descriptor from the current address._x000D_ Firmware writes 1 to enable the DMA. Write 0 is meaningless._x000D_ Read always return 0._x000D_
19	TXQ3_DMA_RUM	**Resume the TX3 queue DMA to operate._x000D_** The DMA will reload the chain descriptor from the current address._x000D_ Firmware writes 1 to enable the DMA. Write 0 is meaningless._x000D_ Read always return 0._x000D_
18	TXQ2_DMA_RUM	**Resume the TX2 queue DMA to operate._x000D_** The DMA will reload the chain descriptor from the current address._x000D_ Firmware writes 1 to enable the DMA. Write 0 is meaningless._x000D_ Read always return 0._x000D_
17	TXQ1_DMA_RUM	**Resume the TX1 queue DMA to operate._x000D_** The DMA will reload the chain descriptor from the current address._x000D_ Firmware writes 1 to enable the DMA. Write 0 is meaningless._x000D_ Read always return 0._x000D_
16	TXQ0_DMA_RUM	**Resume the TX0 queue DMA to operate._x000D_** The DMA will reload the chain descriptor from the current address._x000D_ Firmware writes 1 to enable the DMA. Write 0 is meaningless._x000D_ Read always return 0._x000D_
15	TXQ7_DMA_START	**Start the TX7 queue DMA operation._x000D_** The DMA will load the chain descriptor from the address assigned by HWFTQ7SAR._x000D_ SW must check TXQ7_DMA_STATUS is inactive before start._x000D_

Bit(s)	Name	Description
		Write 0 is meaningless. Read always return 0._x000D_
14	TXQ6_DMA_START	**Start the TX6 queue DMA operation._x000D_** The DMA will load the chain descriptor from the address assigned by HWFTQ6SAR._x000D_ SW must check TXQ6_DMA_STATUS is inactive before start._x000D_ Write 0 is meaningless. Read always return 0._x000D_
13	TXQ5_DMA_START	**Start the TX5 queue DMA operation._x000D_** The DMA will load the chain descriptor from the address assigned by HWFTQ5SAR._x000D_ SW must check TXQ5_DMA_STATUS is inactive before start._x000D_ Write 0 is meaningless. Read always return 0._x000D_
12	TXQ4_DMA_START	**Start the TX4 queue DMA operation._x000D_** The DMA will load the chain descriptor from the address assigned by HWFTQ4SAR._x000D_ SW must check TXQ4_DMA_STATUS is inactive before start._x000D_ Write 0 is meaningless. Read always return 0._x000D_
11	TXQ3_DMA_START	**Start the TX3 queue DMA operation._x000D_** The DMA will load the chain descriptor from the address assigned by HWFTQ3SAR._x000D_ SW must check TXQ3_DMA_STATUS is inactive before start._x000D_ Write 0 is meaningless. Read always return 0._x000D_
10	TXQ2_DMA_START	**Start the TX2 queue DMA operation._x000D_** The DMA will load the chain descriptor from the address assigned by HWFTQ2SAR._x000D_ SW must check TXQ2_DMA_STATUS is inactive before start._x000D_ Write 0 is meaningless. Read always return 0._x000D_
9	TXQ1_DMA_START	**Start the TX1 queue DMA operation._x000D_** The DMA will load the chain descriptor from the address assigned by HWFTQ1SAR._x000D_ SW must check TXQ1_DMA_STATUS is inactive before start._x000D_ Write 0 is meaningless. Read always return 0._x000D_
8	TXQ0_DMA_START	**Start the TX0 queue DMA operation._x000D_** The DMA will load the chain descriptor from the address assigned by HWFTQ0SAR._x000D_ SW must check TXQ0_DMA_STATUS is inactive before start._x000D_

Bit(s)	Name	Description
		Write 0 is meaningless. Read always return 0._x000D_
7	TXQ7_DMA_STOP	**Stop the TX7 queue DMA operation. It will NOT clear the result of TX count set by HWTPCCR(WTSR0/ WTSR1)._x000D_** Firmware writes 1 to stop the DMA. Write 0 is meaningless._x000D_ Read always return current DMA activity (0: stopped, 1: stop command is on-going)._x000D_ If one data port to multiple queues design is configured, any one of these queues stop would lead to these queues stop at the same time. (e.g. If TX queue 1 stops, then TX queue 2, 3, 4, 5, 6, 7 would also be stopped by HW since they share the same data port.)
6	TXQ6_DMA_STOP	**Stop the TX6 queue DMA operation. It will NOT clear the result of TX count set by HWTPCCR(WTSR0/ WTSR1)._x000D_** Firmware writes 1 to stop the DMA. Write 0 is meaningless._x000D_ Read always return current DMA activity (0: stopped, 1: stop command is on-going)._x000D_ If one data port to multiple queues design is configured, any one of these queues stop would lead to these queues stop at the same time. (e.g. If TX queue 1 stops, then TX queue 2, 3, 4, 5, 6, 7 would also be stopped by HW since they share the same data port.)
5	TXQ5_DMA_STOP	**Stop the TX5 queue DMA operation. It will NOT clear the result of TX count set by HWTPCCR(WTSR0/ WTSR1)._x000D_** Firmware writes 1 to stop the DMA. Write 0 is meaningless._x000D_ Read always return current DMA activity (0: stopped, 1: stop command is on-going)._x000D_ If one data port to multiple queues design is configured, any one of these queues stop would lead to these queues stop at the same time. (e.g. If TX queue 1 stops, then TX queue 2, 3, 4, 5, 6, 7 would also be stopped by HW since they share the same data port.)
4	TXQ4_DMA_STOP	**Stop the TX4 queue DMA operation. It will NOT clear the result of TX count set by HWTPCCR(WTSR0/ WTSR1)._x000D_** Firmware writes 1 to stop the DMA. Write 0 is meaningless._x000D_ Read always return current DMA activity (0: stopped, 1: stop command is on-going)._x000D_ If one data port to multiple queues design is configured, any one of these queues stop would lead to these queues stop at the same time. (e.g. If TX queue 1 stops, then TX queue 2, 3, 4, 5, 6, 7 would also be stopped by HW since they share the same data port.)
3	TXQ3_DMA_STOP	**Stop the TX3 queue DMA operation. It will NOT clear the result of TX count set by HWTPCCR(WTSR0/ WTSR1)._x000D_** Firmware writes 1 to stop the DMA. Write 0 is meaningless._x000D_ Read always return current DMA activity (0: stopped, 1: stop command is on-going)._x000D_

Bit(s)	Name	Description
		If one data port to multiple queues design is configured, any one of these queues stop would lead to these queues stop at the same time. (e.g. If TX queue 1 stops, then TX queue 2, 3, 4, 5, 6, 7 would also be stopped by HW since they share the same data port.)
2	TXQ2_DMA_STOP	**Stop the TX2 queue DMA operation. It will NOT clear the result of TX count set by HWTPCCR(WTSR0/ WTSR1)._x000D_** Firmware writes 1 to stop the DMA. Write 0 is meaningless._x000D_ Read always return current DMA activity (0: stopped, 1: stop command is on-going)._x000D_ If one data port to multiple queues design is configured, any one of these queues stop would lead to these queues stop at the same time. (e.g. If TX queue 1 stops, then TX queue 2, 3, 4, 5, 6, 7 would also be stopped by HW since they share the same data port.)
1	TXQ1_DMA_STOP	**Stop the TX1 queue DMA operation. It will NOT clear the result of TX count set by HWTPCCR(WTSR0/ WTSR1)._x000D_** Firmware writes 1 to stop the DMA. Write 0 is meaningless._x000D_ Read always return current DMA activity (0: stopped, 1: stop command is on-going)._x000D_ If one data port to multiple queues design is configured, any one of these queues stop would lead to these queues stop at the same time. (e.g. If TX queue 1 stops, then TX queue 2, 3, 4, 5, 6, 7 would also be stopped by HW since they share the same data port.)
0	TXQ0_DMA_STOP	**Stop the TX0 queue DMA operation. It will NOT clear the result of TX count set by HWTPCCR(WTSR0/ WTSR1)._x000D_** Firmware writes 1 to stop the DMA. Write 0 is meaningless._x000D_ Read always return current DMA activity (0: stopped, 1: stop command is on-going)._x000D_

3813015C HWTPCCR HIF WLAN TX Packet Count Control Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																TQ_CNT_RESET
Type																W1S
Reset																0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TQ_INDEX								INC_TQ_CNT							
Type	WO								WO							

Reset	0	0	0	0					0	0	0	0	0	0	0	0

Bit(s)	Name	Description
16	TQ_CNT_RESET	**Firmware writes 1 to reset the count accumulated for TQ0 ~ TQ7._x000D_** Write 0 is meaningless. Read always return 0._x000D_
15:12	TQ_INDEX	**Firmware writes the TQ index to be increased by setting this field which leads to the same number increased in WTSR.TQX_CNT ._x000D_** Write 0 is meaningless. Read always return 0._x000D_ (X depends on the TQ index)
7:0	INC_TQ_CNT	**Firmware writes the available TQ buffer count by setting this field which leads to the same number increased in WTSR.TQX_CNT ._x000D_** Write 0 is meaningless. Read always return 0._x000D_ (X depends on the TQ index)

38130164 HWFTQ1SAR HIF WLAN Firmware TX Queue 1 Start Address Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	WLAN_TXQ1_DMA_SADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	WLAN_TXQ1_DMA_SADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Bit(s)	Name	Description
31:2	WLAN_TXQ1_DMA_SADDR	**The start address of buffer chain of TX1 queue in unit of DW._x000D_**

38130180 HWFRQ0SAR HIF WLAN Firmware RX Queue 0 Start Address Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	WLAN_RXQ0_DMA_SADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Name	WLAN_RXQ0_DMA_SADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Bit(s)	Name	Description
31:2	WLAN_RXQ0_DMA_SADDR	**The start address of buffer chain of RX0 queue in unit of DW._x000D_**

38130184 HWFRQ1SAR HIF WLAN Firmware RX Queue 1 Start Address Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	WLAN_RXQ1_DMA_SADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	WLAN_RXQ1_DMA_SADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Bit(s)	Name	Description
31:2	WLAN_RXQ1_DMA_SADDR	**The start address of buffer chain of RX1 queue in unit of DW.**

381301A0 H2DRM0R Host to Device Receive Mailbox 0 Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	H2D_RM0															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	H2D_RM0															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	H2D_RM0	**This register is used by firmware to receive data from SDIO controller, which is updated through H2DSM0R by host driver._x000D_**

381301A4 H2DRM1R Host to Device Receive Mailbox 1 Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	H2D_RM1															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	H2D_RM1															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	H2D_RM1	**This register is used by firmware to receive data from SDIO controller, which is updated through H2DSM1R by host driver._x000D_**

381301A8 D2HSM0R Device to Host Send Mailbox 0 Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	D2H_SM0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	D2H_SM0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	D2H_SM0	**This register is used by firmware to transmit data to SDIO controller, it will be updated to D2HRM0R and read by host driver._x000D_**

Bit(s)	Name	Description

381301AC D2HSM1R Device to Host Send Mailbox 1 Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	D2H_SM1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	D2H_SM1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	D2H_SM1	**This register is used by firmware to transmit data to SDIO controller, it will be updated to D2HRM1R and read by host driver._x000D_**

381301C0 HWRQ0CR HIF WLAN RX Queue 0 Control Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name													RXQ0_DMA_STATUS	RXQ0_DMA_RUM	RXQ0_DMA_START	RXQ0_DMA_STOP
Type													RO	W1S	W1S	W1S
Reset													0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RXQ0_PACKET_LENGTH															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
19	RXQ0_DMA_STATUS	**Read for the RX0 queue DMA status._x000D_** When the SDIO Controller is reset, the queue is in the inactive state by default. After receiving a START or RESUME command and executing it without error, the queue enters the active state. When the queue is

Bit(s)	Name	Description
		empty or stopped by a STOP command, it returns to the inactive state._x000D_ 0: inactive_x000D_ 1: active_x000D_
18	RXQ0_DMA_RUM	**Resume the RX0 queue DMA to operate._x000D_** The DMA will reload the chain descriptor from the current address._x000D_ Firmware writes 1 to enable the DMA. Write 0 is meaningless._x000D_ Read always return 0._x000D_
17	RXQ0_DMA_START	**Start the RX0 queue DMA operation._x000D_** The DMA will load the chain descriptor from the address assigned by HWFRQ0SAR._x000D_ SW must check RXQ0_DMA_STATUS is inactive before start._x000D_ Write 0 is meaningless. Read always return 0._x000D_
16	RXQ0_DMA_STOP	**Stop the RX0 queue DMA operation, the content in the RXQ0 FIFO and RXQ0 length FIFO will be cleared._x000D_** Firmware writes 1 to stop the DMA. Write 0 is meaningless._x000D_ Read return current RXQ0 operation state (1: stop operation is on-going, 0: stop operation is finished)._x000D_
15:0	RXQ0_PACKET_LENGTH	**When write:_x000D_** To indicate HIF that 1 RX packet in this packet length is queued into this RX queue._x000D_ When read:_x000D_ Read the 1st RX packet length indicated from this queue, and will be 0 when queue is empty._x000D_ _x000D_ FW will write this FIFO-like port (at most 64 entries depends on the hardware configuration for each project) together with RXQ1_DMA_RUM bit been set, after RX packet is queued into descriptor chain. _x000D_ RX packet with length been set by this field is able to be read by host driver, which is through reading WRPLR, or INT enhance mode, or RX enhance mode._x000D_ None-empty entry will generate RX done interrupt, and corresponding entry will be cleared by HW after this packet length is read by host driver._x000D_ Write 0 is meaning-less._x000D_ 0: inactive 1: active

Bit(s)	Name	Description

381301C4 HWRQ1CR HIF WLAN RX Queue 1 Control Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name													RXQ1_DMA_STATUS	RXQ1_DMA_RUM	RXQ1_DMA_START	RXQ1_DMA_STOP
Type													RO	W1S	W1S	W1S
Reset													0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RX1_PACKET_LENGTH															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
19	RXQ1_DMA_STATUS	**Read for the RXQ1 DMA status._x000D_** When the SDIO Controller is reset, the queue is in the inactive state by default. After receiving a START or RESUME command and executing it without error, the queue enters the active state. When the queue is empty or stopped by a STOP command, it returns to the inactive state._x000D_ 0: inactive 1: active_x000D_
18	RXQ1_DMA_RUM	**Resume the RX1 queue DMA to operate._x000D_** The DMA will reload the chain descriptor from the current address._x000D_ Firmware writes 1 to enable the DMA. Write 0 is meaningless._x000D_ Read always return 0._x000D_
17	RXQ1_DMA_START	**Start the RX1 queue DMA operation._x000D_** The DMA will load the chain descriptor from the address assigned by HWFRQ1SAR._x000D_ SW must check RXQ1_DMA_STATUS is inactive before start._x000D_ Write 0 is meaningless. Read always return 0._x000D_
16	RXQ1_DMA_STOP	**Stop the RX1 queue DMA operation, the content in the RXQ1 FIFO and RXQ1 length FIFO will be cleared._x000D_**

Bit(s)	Name	Description
		Firmware writes 1 to stop the DMA. Write 0 is meaningless._x000D_ Read return current RXQ1 operation state (1: active, 0: stopped)._x000D_
15:0	RX1_PACKET_LENGTH	**When write:_x000D_** To indicate HIF that 1 RX packet in this packet length is queued into this RX queue._x000D_ When read:_x000D_ Read the 1st RX packet length indicated from this queue, and will be 0 when queue is empty._x000D_ _x000D_ FW will write this FIFO-like port (at most 64 entries depends on the hardware configuration for each project) together with RXQ1_DMA_RUM bit been set, after RX packet is queued into descriptor chain. _x000D_ RX packet with length been set by this field is able to be read by host driver, which is through reading WRPLR, or INT enhance mode, or RX enhance mode._x000D_ None-empty entry will generate RX done interrupt, and corresponding entry will be cleared by HW after this packet length is read by host driver._x000D_ Write 0 is meaning-less._x000D_

381301EC HWFIOCDR HIF WLAN Firmware GPD IOC bit Disable Register 00000FFF

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					RXQ3_IOC_DIS	RXQ2_IOC_DIS	RXQ1_IOC_DIS	RXQ0_IOC_DIS	TXQ7_IOC_DIS	TXQ6_IOC_DIS	TXQ5_IOC_DIS	TXQ4_IOC_DIS	TXQ3_IOC_DIS	TXQ2_IOC_DIS	TXQ1_IOC_DIS	TXQ0_IOC_DIS
Type					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset					1	1	1	1	1	1	1	1	1	1	1	1

Bit(s)	Name	Description
11	RXQ3_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFRE1SR. 0: Enable IOC function.

Bit(s)	Name	Description
		1: Disable IOC function (always issue interrupt when GPD done)
10	RXQ2_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFRE1SR. 0: Enable IOC function. 1: Disable IOC function (always issue interrupt when GPD done)
9	RXQ1_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFRE1SR. 0: Enable IOC function. 1: Disable IOC function (always issue interrupt when GPD done)
8	RXQ0_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFRE1SR. 0: Enable IOC function. 1: Disable IOC function (always issue interrupt when GPD done)
7	TXQ7_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFTE0SR. 0: Enable IOC function. 1: Disable IOC function (always issue interrupt when GPD done)
6	TXQ6_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFTE0SR. 0: Enable IOC function. 1: Disable IOC function (always issue interrupt when GPD done)
5	TXQ5_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.**

Bit(s)	Name	Description
		If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFTE0SR. 0: Enable IOC function. 1: Disable IOC function (always issue interrupt when GPD done)
4	TXQ4_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFTE0SR. 0: Enable IOC function. 1: Disable IOC function (always issue interrupt when GPD done)
3	TXQ3_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFTE0SR. 0: Enable IOC function. 1: Disable IOC function (always issue interrupt when GPD done)
2	TXQ2_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFTE0SR. 0: Enable IOC function. 1: Disable IOC function (always issue interrupt when GPD done)
1	TXQ1_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFTE0SR. 0: Enable IOC function. 1: Disable IOC function (always issue interrupt when GPD done)
0	TXQ0_IOC_DIS	**If firmware write 1 to this register, the corresponding queue always issue interrupt event when GPD is done.** If firmware write 0 to this register, the corresponding queue will issue interrupt event base on GPD IOC bit. If current GPD IOC = 1, GPD done interrupt event will be issued and latched into HWFTE0SR. 0: Enable IOC function.

Bit(s)	Name	Description
		1: Disable IOC function (always issue interrupt when GPD done)

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this "Document") is subject to your (including the corporation or other legal entity you represent, collectively "You") acceptance of the terms and conditions set forth below ("T&C"). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don't agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively "MediaTek") or its licensors and is provided solely for Your internal use with MediaTek's chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek's suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek's product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.