ASR582X Series

Peripheral Application Notes

Version 1.3.0

Issue Date 2022-11-24

About This Document

This document provides the detailed descriptions of the peripheral API interface provided by the ASR582X series chips.

Intended Readers

This document is mainly for engineers who use this chip to develop their own platform and products, for instance:

- PCB Hardware Development Engineer
- Software Engineer
- Technical Support Engineer

Included Chip Models

The product models corresponding to this document are ASR582X series Wi-Fi+BLE Combo SoC chip.

Copyright Notice

Trademark Statement

Disclaimer

ASR does not give any warranty of any kind and may make improvements and/or changes in this document or in the product described in this document at any time.

This document is only used as a guide, and no contents in the document constitute any form of warranty. Information in this document is subject to change without notice.

All liability, including liability for infringement of any proprietary rights caused by using the information in this document, is disclaimed.

ASR Microelectronics Co., Ltd.

Address: 9F, Building 10, No. 399 Keyuan Road, Zhangjiang High-tech Park, Pudong New Area, Shanghai, 201203, China

Homepage: http://www.asrmicro.com/

Revision History

Date	Version	Release Notes
2020.10	V1.0.0	First release.
2020.11	V1.1.0	Updated GPIO API Description.
2020.12	V1.2.0	Added ADC API Description.
2021.01	V1.3.0	Added I2S/RAM Layout/PSRAM/AES API Description.

Table of Contents

List of Tables

1. GPIO

1.1 Functional Description

ASR582X (48 PIN) supports a total of 20 GPIOs, with the functions of input, output, pull control and interrupt. Structure **duet_gpio_dev_t** is used to describe ASR582X GPIO hardware, and the definition is shown below:

Table 1-1 duet_gpio_dev_t Structure

Member Variable Name	Type	Description
port	uint8_t	GPIO index, between 0 to 19
config	duet_gpio_config_t	GPIO mode configuration: ● DUET_ANALOG_MODE ● DUET_IRQ_MODE ● DUET_OUTPUT_PUSH_PULL ● DUET_OUTPUT_OPEN_DRAIN_PULL_UP ● DUET_OUTPUT_OPEN_DRAIN_NO_PULL ● DUET_INPUT_PULL_DOWN ● DUET_INPUT_PULL_UP ● DUET_INPUT_HIGH_IMPEDANCE
priv	void *	user-defined private data

1.2 APIs

1.2.1 duet_gpio_init

Function	Initialize a GPIO pin, and prepares a GPIO pin for use
Parameters	● gpio: the GPIO device which should be initialized
Return	Result: 0: success EIO: if an error occurs in any step

1.2.2 duet_gpio_output_high

Function	Set an output GPIO pin high
Parameters	● gpio: the GPIO device which should be set high
Return	Result: 0: success EIO: if an error occurs in any step

1.2.3 duet_gpio_output_low

Function	Set an output GPIO pin low
Parameters	● gpio: the GPIO device which should be set low
Return	Result: 0: success EIO: if an error occurs in any step

1.2.4 duet_gpio_output_toggle

Function	Toggle the voltage of an output GPIO pin
Parameters	● gpio: the GPIO device which should toggle
Return	Result: 0: success EIO: if an error occurs in any step

1.2.5 duet_gpio_input_get

Function	Get the state of an input GPIO pin
Parameters	● gpio: the GPIO device which should be read ● value: used to store GPIO state
Return	Result: 0: success EIO: if an error occurs in any step

1.2.6 duet_gpio_enable_irq

Function	Enable an interrupt trigger for an input GPIO pin
Parameters	● gpio: the GPIO device which will provide the interrupt trigger ● trigger: the type of trigger (rising/falling edge or both) ● handler: a function pointer to the interrupt handler ● arg: an argument that will be passed to the interrupt handler
Return	Result: ● 0: success ● EIO: if an error occurs in any step

1.2.7 duet_gpio_disable_irq

Function	Disable an interrupt trigger for an input GPIO pin
Parameters	● gpio: the GPIO device which will provide the interrupt trigger
Return	Result: ● 0: success ● EIO: if an error occurs in any step

1.2.8 duet_gpio_clear_irq

Function	Clear interrupt flag for an input GPIO pin
Parameters	● gpio: the GPIO device which will provide the interrupt trigger
Return	Result: ● 0: success ● EIO: if an error occurs in any step

1.2.9 duet_gpio_finalize

Function	Set a GPIO pin in default state
Parameters	● gpio: the GPIO device which should be de-initialized
Return	Result: ● 0: success ● EIO: if an error occurs in any step

2. Watchdog

2.1 Functional Description

The watchdog module applies a reset to a system in the event of a software failure, providing a way to recover from software crashes. Structure **duet_wdg_dev_t** is used to describe ASR582X watchdog hardware, and the definition is shown below:

Table 2-1 duet_wdg_dev_t structure

Member Variable Name	Type	Description
port	uint8_t	watchdog index, which should always be 0
config	duet_wdg_config_t	watchdog configuration: ● timeout: watchdog timeout configuration
priv	void *	user-defined private data

2.2 APIs

2.2.1 duet_wdg_init

Function	Initialize the watchdog
Parameters	● wdg: the watchdog device which should be initialized
Return	Result: 0: success EIO: if an error occurs in any step

2.2.2 duet_wdg_reload

Function	Reload the watchdog counter
Parameters	● wdg: the watchdog device
Return	Result: 0: success EIO: if an error occurs in any step

2.2.3 duet_wdg_start

Function	Restart hardware watchdog
Parameters	● N/A
Return	N/A

2.2.4 duet_wdg_stop

Function	Stop hardware watchdog
Parameters	● N/A
Return	N/A

2.2.5 duet_wdg_finalize

Function	De-initialize the watchdog
Parameters	● wdg: the watchdog device
Return	Result: 0: success EIO: if an error occurs in any step

3. Timer

3.1 Functional description

ASR582X supports two timers whose function and operation are identical. For each timer, the following modes of operation are available:

Periodic timer mode: The counter generates an interrupt at a constant interval, reloading the original value after wrapping past zero.

One-shot timer mode: The counter generates an interrupt once. When the counter reaches 0, it halts until you reprogram it.

Structure **duet_timer_dev_t** is used to describe the ASR582X timer hardware, and the definition is shown below:

Table 3-1 duet_timer_dev_t structure

Member Variable Name	Type	Description
port	uint8_t	timer index, which should be 0 or 1
config	duet_timer_config_t	timer configuration: ● period: unit in us ● reload_mode: ➢ TIMER_RELOAD_AUTO: the timer reloads automatically ➢ TIMER_RELOAD_MANU: the timer reloads manually ● cb: the callback function for timer timeout ● arg: arguments passed to the callback function
priv	void *	user-defined private data

3.2 APIs

3.2.1 duet_timer_init

Function	Initialize a hardware timer
Parameters	● tim: the timer device
Return	Result: 0: success EIO: if an error occurs in any step

3.2.2 duet_timer_start

Function	Start a hardware timer
Parameters	● tim: the timer device
Return	Result: 0: success EIO: if an error occurs in any step

3.2.3 duet_timer_stop

Function	Stop a hardware timer
Parameters	● tim: the timer device
Return	Result: 0: success EIO: if an error occurs in any step

3.2.4 duet_timer_get

Function	Get hardware timer remaining time
Parameters	● tim: the timer device
Return	Result: -1: if an error occurs in any step Others: timer remaining time

3.2.5 duet_timer_reload

Function	Reload the hardware timer value
Parameters	● tim: the timer device
Return	Result: 0: success -1: if an error occurs in any step

3.2.6 duet_timer_finalize

Function	De-initialize a TIMER interface, and turn off a TIMER hardware interface
Parameters	● tim: the timer device
Return	Result: 0: success EIO: if an error occurs in any step

4. RTC

4.1 Functional Description

There is only one RTC (real time clock) hardware in ASR582X, which is used to provide accurate time information even when ASR582X enters the deep-sleep mode. Structure **duet_rtc_dev_t** is used to describe ASR582X RTC hardware, and the definition is shown below:

Table 4-1 duet_rtc_dev_t structure

Member Variable Name	Type	Description
port	uint8_t	RTC index, must be 0
config	duet_rtc_config_t	timer configuration: ● format: time format DEC or BCD
priv	void *	user-defined private data

4.2 APIs

4.2.1 duet_rtc_init

Function	This function will initialize the on-board CPU real time clock
Parameters	● rtc: the RTC device
Return	Result: 0: success EIO: if an error occurs in any step

4.2.2 duet_rtc_get_time

Function	This function will return the value of time read from the on-board real time clock
Parameters	● rtc: the RTC device ● time: pointer to a time structure
Return	Result: 0: success EIO: if an error occurs in any step

4.2.3 duet_rtc_set_time

Function	This function will set RTC time to a new value
Parameters	● rtc: the RTC device ● time: pointer to a time structure
Return	Result: 0: success EIO: if an error occurs in any step

4.2.4 duet_rtc_finalize

Function	This function will finalize the on-board real time clock
Parameters	● rtc: the RTC device
Return	Result: 0: success EIO: if an error occurs in any step

5. PWM

5.1 Functional Description

There are a total of eight PWM (Pulse Width Modulation) channels in ASR582X, which can output square waveform with specific frequency and duty cycle. Structure **duet_pwm_dev_t** is used to describe ASR582X PWM hardware, and the definition is shown below:

Table 5-1 duet_pwm_dev_t structure

Member Variable Name	Type	Description
port	uint8_t	PWM channel index, which should be between 0 to 7
config	duet_pwm_config_t	PWM configuration: ● duty_cycle: the PWM duty cycle ● freq: the PWM frequency
priv	void *	user-defined private data

5.2 APIs

5.2.1 duet_pwm_init

Function	Initialize a PWM pin
Parameters	● pwm: the PWM device
Return	Result: 0: success EIO: if an error occurs in any step

5.2.2 duet_pwm_start

Function	Start Pulse-Width Modulation signal output on a PWM pin
Parameters	● pwm: the PWM device
Return	Result: 0: success EIO: if an error occurs in any step

5.2.3 duet_pwm_stop

Function	Stop output on a PWM pin
Parameters	● pwm: the PWM device
Return	Result: 0: success EIO: if an error occurs in any step

5.2.4 duet_pwm_para_chg

Function	Change the para of PWM
Parameters	● pwm: the PWM device ● para: the para of PWM
Return	Result: 0: success EIO: if an error occurs in any step

5.2.5 duet_pwm_finalize

Function	De-initialize an PWM interface, and turn off an PWM hardware interface
Parameters	● pwm: the PWM device
Return	Result: 0: success EIO: if an error occurs in any step

6. Flash

6.1 Functional Description

ASR582X has one embedded SPI NOR Flash with XIP function. Structure **duet_logic_partition_t** is used to describe Flash memory mapping, and the definition is shown below:

Table 6-1 duet_logic_partition_t structure

Member Variable Name	Type	Description
partition_owner	duet_flash_t	Flash partition owner: ● FLASH_EMBEDDED ● FLASH_SPI ● FLASH_QSPI ● FLASH_MAX ● FLASH_NONE
partition_description	char *	Flash partition description string
partition_start_addr	uint32_t	Flash partition start address
partition_length	uint32_t	Flash partition length in bytes
partition_options	uint32_t	Flash partition read and write permission

6.2 APIs

6.2.1 duet_flash_init

Function	Initialize a FLASH interface
Parameters	● N/A
Return	Result: 0: success EIO: if an error occurs in any step

6.2.2 duet_flash_get_infok

Function	Get the information of the specified flash area
Parameters	● in_partition: the target flash logical partition
Return	duet_logic_partition_t struct pointer

6.2.3 duet_flash_erase

Function	Erase an area on a flash logical partition
Parameters	● in_partition: the target flash logical partition which should be erased ● off_set: start address of the erased flash area ● size: size of the erased flash area
Return	Result: 0: success EIO: if an error occurs in any step

6.2.4 duet_flash_write

Function	Write data to an area on a flash logical partition without erasing it
Parameters	● in_partition: the target flash logical partition which should be written ● off_set: point to the start address that the data is written to, and point to the last unwritten address after this function is returned, so you can call this function several times without updating this start address. ● in_buf: point to the data buffer that will be written to flash ● in_buf_len: the length of the buffer
Return	Result: 0: success EIO: if an error occurs in any step

6.2.5 duet_flash_erase_write

Function	First erase an area on a flash logical partition, and then write data to it
Parameters	● in_partition: the target flash logical partition which should be written ● off_set: point to the start address that the data is written to, and point to the last unwritten address after this function is returned, so you can call this function several times without updating this start address. ● in_buf: point to the data buffer that will be written to flash ● in_buf_len: the length of the buffer
Return	Result: 0: success EIO: if an error occurs in any step

6.2.6 duet_flash_read

Function	Read data from an area on a flash to data buffer in RAM
Parameters	● in_partition: the target flash logical partition which should be read ● off_set: point to the start address that the data is read, and point to the last unread address after this function is returned, so you can call this function several times without updating this start address. ● out_buf: point to the data buffer that stores the data read from flash ● in_buf_len: the length of the buffer
Return	Result: 0: success EIO : if an error occurs in any step

6.2.7 duet_flash_enable_secure

Function	Set security options on a logical partition
Parameters	● partition: the target flash logical partition ● offset: point to the start address where secure should be enabled, and point to the last non-secure address after this function is returned, so you can call this function several times without updating this start address ● size: size of enabled flash area
Return	Result: 0: success EIO: if an error occurs in any step

6.2.8 duet_flash_dis_secure

Function	Disable security options on a logical partition
Parameters	● partition: the target flash logical partition ● offset: point to the start address where secure should be disabled, and the offset increases automatically, so you can call this function several times without updating this start address ● size: size of disabled flash area
Return	Result: 0: success EIO: if an error occurs in any step

7. UART

7.1 Functional Description

There are three identical UART devices in ASR582X, which support 5-8 data bits per character, optional parity bit and 1, 1.5 or 2 stop bits. Structure **duet_uart_dev_t** is used to describe ASR582X UART hardware, and the definition is shown below:

Table 7-1 duet_uart_dev_t structure

Member Variable Name	Type	Description
port	uint8_t	UART device index, which must be 0, 1 or 2
config	duet_uart_config_t	UART configuration: ● baud_rate: UART baud rate ● data_width: UART data bits per character DATA_5BIT DATA_6BIT DATA_7BIT DATA_8BIT ● parity: UART parity bit PARITY_NO PARITY_ODD PARITY_EVEN ● stop_bits: UART stop bits STOP_1BIT STOP_2BITS ● flow_control: UART flow control FLOW_CTRL_DISABLED FLOW_CTRL_CTS FLOW_CTRL_RTS FLOW_CTRL_CTS_RTS ● mode: UART mode TX_MODE RX_MODE TX_RX_MODE
priv	void *	user-defined private data

7.2 APIs

7.2.1 duet_uart_struct_init

Function	Initialize a UART interface struct as default value baud_rate: 115200 date_width: 8 bits Parity: no Stop bit: 1bit flow_control: disable mode: tx_rx_mode
Parameters	● UART_InitStruct: the interface which should be inited
Return	Result: 0: success EIO: if an error occurs in any step

7.2.2 duet_uart_init

Function	Initialize a UART interface
Parameters	● uart: the interface which should be initialized
Return	Result: 0: success EIO: if an error occurs in any step

7.2.3 duet_uart_send

Function	Transmit data on a UART interface
Parameters	● uart: the UART interface ● data: pointer to the start of data ● size: number of bytes to transmit ● timeout: timeout in millisecond, not use, reserved for future
Return	Result: 0: success EIO: if an error occurs in any step

7.2.4 duet_uart_finalize

Function	De-initialize a UART interface
Parameters	● uart: the interface which should be de-initialized
Return	Result: 0: success EIO: if an error occurs in any step

7.2.5 duet_uart_dma_config

Function	Enable UART TX/RX DMA support
Parameters	● uart: the UART interface ● dma_tx_rx_sel: set TX or RX DMA ● new_state: enable or disable DMA
Return	Result: 0: success EIO: if an error occurs in any step

7.2.6 getUartxViaIdx

Function	Get UART interface addr
Parameters	● uart_idx: the UART index
Return	Result: NULL: if an error occurs in any step Others: UART interface addr

7.2.7 duet_uart_get_flag_status

Function	Get UART flag status
Parameters	● UARTx: UART interface addr ● uart_flag: UART status flag
Return	Result: SET: UART flag set RESET: UART flag not set

7.2.8 duet_uart_interrupt_config

Function	Set UART interrupt mode
Parameters	● UARTx: UART interface addr ● uart_int: UART interrupt mode ● new_state: UART interrupt enable/disable
Return	N/A

7.2.9 duet_uart_get_interrupt_status

Function	Get UART interrupt status
Parameters	● UARTx: UART interface addr ● uart_interrupt: UART interrupt flag
Return	Result: SET: corresponding UART interrupt occurred RESET: no corresponding UART interrupt occurred

7.2.10 duet_uart_clear_interrupt

Function	Clear UART interrupt status
Parameters	● UARTx: UART interface addr ● uart_interrupt: UART interrupt flag
Return	Result: 0 : success

7.2.11 duet_uart_set_rx_fifo_threshold

Function	Set UART RX FIFO threshold
Parameters	● UARTx: UART interface addr ● uart_fifo_level: UART RX FIFO threshold
Return	Result: 0 : success

7.2.12 duet_uart_set_tx_fifo_threshold

Function	Set UART TX FIFO threshold
Parameters	● UARTx: UART interface addr ● uart_fifo_level: UART RX FIFO threshold
Return	Result: 0 : success

7.2.13 duet_uart_start

Function	Restart UART interface
Parameters	● UARTx: UART interface addr
Return	N/A

7.2.14 duet_uart_stop

Function	Stop UART interface
Parameters	● UARTx: UART interface addr
Return	N/A

7.2.15 UART_SendData

Function	Send data
Parameters	● UARTx: the interface which should be used to send data ● Data: character to be send
Return	Result: 0: success EIO: if an error occurs in any step

7.2.16 UART_ReceiveData

Function	Receive data, wait when FIFO is not empty, and get character from FIFO
Parameters	● UARTx: the interface which should be used to receive data
Return	Result: Receive data value

8. I2C

8.1 Functional Description

There are two identical I2C devices in ASR582X, which can be configured as master or slave. They also support two speed modes and 7-bit and 10-bit addressing mode. Structure **duet_i2c_dev_t** is used to describe ASR582X I2C hardware, and the definition is shown below:

Table 8-1 duet_i2c_dev_t structure

Member Variable Name	Type	Description
port	uint8_t	I2C index, which must be 0 or 1
config	duet_i2c_config_t	I2C configuration: ● address_width: 7-bit or 10-bit address width ● freq: the I2C SCL frequency ● mode: I2C master mode or slave mode ● I2C_MODE_MASTER ● I2C_MODE_SLAVE ● dev_addr: I2C slave address
priv	void *	user-defined private data

8.2 APIs

8.2.1 duet_i2c_init

Function	Initialize an I2C interface, and prepares an I2C hardware interface for communication as a master or slave
Parameters	● i2c: the device for which the I2C port should be initialized
Return	Result: 0: success EIO: if an error occurs in any step

8.2.2 duet_i2c_master_send

Function	I2C master sends data to the specific device address
Parameters	● i2c: the I2C device ● dev_addr: device address ● data: data buffer to be sent ● size: data size ● timeout: timeout in millisecond
Return	Result: 0: success EIO: if an error occurs in any step

8.2.3 duet_i2c_master_recv

Function	I2C master receives data from the specific device address
Parameters	● i2c: the I2C device ● dev_addr: device address ● data: buffer for storing data received ● size: data size ● timeout: timeout in millisecond
Return	Result: 0: success EIO: if an error occurs in any step

8.2.4 duet_i2c_master_repeated_write_read

Function	I2C master sends data to the specific device address and receives data from the specific device address
Parameters	● I2Cx: I2C interface addr ● slave_addr: slave device address ● pwdata: data buffer to be sent ● rdata: buffer for storing data received ● wlen: pwdata size

	● rlen: rdata size ● timeout: timeout in millisecond
Return	Result: 0: success EIO: if an error occurs in any step

8.2.5 duet_i2c_mem_write

Function	I2C master writes data to memory (e.g. EEPROM)
Parameters	● i2c: the I2C device ● dev_addr: device address ● mem_addr: memory start address where data is written ● mem_addr_size: memory address size ● data: data buffer to write to ● size: data size ● timeout: timeout in millisecond
Return	Result: 0: success EIO: if an error occurs in any step

8.2.6 duet_i2c_mem_read

Function	I2C master reads data from memory (e.g. EEPROM)
Parameters	● i2c: the I2C device ● dev_addr: device address ● mem_addr: memory start address where data is read ● mem_addr_size: memory address size ● data: buffer for storing data read from memory ● size: data size ● timeout: timeout in millisecond
Return	Result: 0: success EIO: if an error occurs in any step

8.2.7 duet_i2c_master_dma_send

Function	I2C master sends data to the specific device address using DMA
Parameters	● iic_idx: the I2C port ● data: data buffer to be sent ● len: data size
Return	N/A

8.2.8 duet_i2c_master_dma_recv

Function	I2C master receives data to the specific device address using DMA
Parameters	● iic_idx: the I2C port ● data: buffer for storing data received ● len: data size
Return	N/A

8.2.9 duet_i2c_finalize

Function	De-initialize an I2C device
Parameters	● i2c: the I2C device
Return	Result: 0: success EIO: if an error occurs in any step

8.2.10 i2c_write_byte_cmd

Function	I2C writes one byte command
Parameters	● I2Cx: I2C interface addr ● data: one byte to be sent
Return	N/A

8.2.11 i2c_read_byte_cmd

Function	I2C reads one byte command
Parameters	● I2Cx: I2C interface addr
Return	N/A

8.2.12 i2c_write_byte

Function	I2C writes one byte to FIFO or buffer register
Parameters	● I2Cx: I2C interface addr ● data: byte to be sent
Return	N/A

8.2.13 i2c_receive_byte

Function	I2C reads one byte from FIFO or buffer register
Parameters	● I2Cx: I2C interface addr
Return	Result: the byte reads from FIFO or buffer register

8.2.14 i2c_clear_interrupt

Function	Clear I2C interrupt status
Parameters	● I2Cx: I2C interface addr ● I2C_INTR: I2C interrupt flag
Return	N/A

8.2.15 i2c_set_tb

Function	I2C sets TB for transmitting and receiving a byte
Parameters	● I2Cx: I2C interface addr
Return	N/A

9. I2S

9.1 Functional Description

Structure **duet_i2s_dev_t** is used to describe ASR582X I2S hardware, and the definition is shown below:

Table 9-1 duet_i2s_dev_t structure

Member Variable Name	Type	Description
config	duet_i2s_config_t	I2S configuration: ● i2s_sample_rate: I2S sample rate ● i2s_mclk_src: I2S mclk's clock source ● i2s_ws: i2s_lck ● i2s_role: I2S role ➢ I2S_ROLE_MASTER ➢ I2S_ROLE_SALVE ● i2s_word_size: I2S word size ● i2s_tx_en ➢ ENABLE ➢ DISABLE ● i2s_rx_en ➢ ENABLE ➢ DISABLE ● i2s_fifo_threshold ● i2s_mode
priv	void *	user-defined private data

9.2 APIs

9.2.1 duet_i2s_init

Function	Initialize an I2S interface, and configure I2S communication
Parameters	● I2Sx: I2S interface addr ● pI2S_struct: I2S init struct
Return	Result: 0: success EIO: if an error occurs in any step

9.2.2 duet_i2s_struct_init

Function	Initialize an I2S interface struct as default value i2s_role: master i2s_word_size: 16bit i2s_tx_en: enable i2s_rx_en: enable i2s_fifo_threshold: level4 i2s_sample_rate: 44100 i2s_mclk_src: 120000000 i2s_ws:16 i2s_mode: Philips
Parameters	● PI2S_struct: the interface which should be inited
Return	N/A

9.2.3 i2s_get_interrupt_status

Function	Get I2S interrupt status
Parameters	● I2Sx: I2S interface addr ● i2s_interrupt: I2S interrupt flag
Return	Result: SET: corresponding I2S interrupt occurred RESET: no corresponding I2S interrupt occurred

9.2.4 duet_i2s_interrupt_config

Function	Set I2S interrupt mode
Parameters	● I2Sx: I2S interface addr ● i2s_interrupt: I2S interrupt mode ● new_state: I2S interrupt enable/disable ➢ ENABLE ➢ DISABLE
Return	N/A

9.2.5 duet_i2s_interrupt_clear

Function	Clear I2S interrupt status
Parameters	● I2Sx: I2S interface addr ● i2s_interrupt: I2S interrupt flag
Return	N/A

9.2.6 duet_i2s_cmd

Function	Open or close I2S function
Parameters	● I2Sx: I2S interface addr ● new_state: I2S function enable/disable ➢ ENABLE ➢ DISABLE
Return	N/A

9.2.7 duet_i2s_tx_block_cmd

Function	Open or close I2S TX block
Parameters	● I2Sx: I2S interface addr ● new_state: I2S TX block enable/disable ➢ ENABLE ➢ DISABLE
Return	N/A

9.2.8 duet_i2s_rx_block_cmd

Function	Open or close I2S RX block
Parameters	● I2Sx: I2S interface addr ● new_state: I2S RX block enable/disable ➢ ENABLE ➢ DISABLE
Return	N/A

9.2.9 duet_i2s_tx_channel_cmd

Function	Open or close I2S TX channel
Parameters	● I2Sx: I2S interface addr ● new_state: I2S TX channel enable/disable ➢ ENABLE ➢ DISABLE
Return	N/A

9.2.10 duet_i2s_rx_channel_cmd

Function	Open or close I2S RX channel
Parameters	● I2Sx: I2S interface addr ● new_state: I2S RX channel enable/disable ➢ ENABLE ➢ DISABLE
Return	N/A

9.2.11 duet_i2s_master_clock_cmd

Function	Open or close I2S clock source
Parameters	● I2Sx: I2S interface addr ● new_state: I2S clock source enable/disable ➢ ENABLE ➢ DISABLE
Return	N/A

9.2.12 duet_i2s_send_data

Function	I2S device sends the specified sized data
Parameters	● I2Sx: I2S interface addr ● left_chan_data: left channel data ● right_chan_data: right channel data ● len: data size
Return	N/A

9.2.13 duet_i2s_receive_data

Function	I2S device receives the specified sized data
Parameters	● I2Sx: I2S interface addr ● lr: channel select (left or right)
Return	Result: the word I2S received

10. SPI

10.1 Functional Description

There are three SPI devices in ASR582X, and software can change the SPI's role. There are four dedicated lines (clk, cs, spitx, spirx) for each SPI controller, all of which support the Motorola SPI standard. Structure **duet_spi_dev_t** is used to describe ASR582X SPI hardware, and the definition is shown below:

Table 10-1 duet_spi_dev_t structure

Member Variable Name	Type	Description
port	uint8_t	SPI index, which must be 0, 1, 2
config	duet_spi_config_t	SPI configuration: ● mode: SPI mode, setting SPI as master or slave ● freq: the SPI clk frequency
priv	void *	user-defined private data

10.2 APIs

10.2.1 duet_spi_init

Function	Initialize an SPI interface, and configure the SPI communication
Parameters	● spi: the device for which the SPI port should be initialized
Return	Result: 0: success EIO: if an error occurs in any step

10.2.2 duet_spi_struct_init

Function	Initialize a SPI interface struct as default value freq:1000000 mode: master mode
Parameters	● init_struct: the interface which should be inited
Return	N/A

10.2.3 getSpixVialdx

Function	Get SPI interface addr
Parameters	● spi_idx: the SPI index
Return	Result: NULL: if an error occurs in any step Others: SPI interface addr

10.2.4 duet_spi_interrupt_config

Function	Set SPI interrupt mode
Parameters	● SPIx: SPI interface addr ● spi_interrupt: SPI interrupt mode ● new_state: SPI interrupt enable/disable
Return	N/A

10.2.5 duet_spi_dma_config

Function	Enable SPI TX/RX DMA support
Parameters	● spi: the SPI interface ● dma_tx_rx_sel: set TX or RX DMA ● new_state: enable or disable DMA
Return	Result: 0: success EIO: if an error occurs in any step

10.2.6 duet_spi_cmd

Function	Enable/disable SPI interface
Parameters	● SPIx: SPI interface addr ● new_state: enable or disable SPI
Return	N/A

10.2.7 duet_spi_cpol_cpha_config

Function	Spi cpol cpha mode config
Parameters	● spi: the SPI interface ● mode: spi cpol cpha mode
Return	Result: 0: success EIO: if an error occurs in any step

10.2.8 duet_spi_get_flag_status

Function	Get spi flag status
Parameters	● SPIx: SPI interface addr ● spi_flag: SPI status flag
Return	Result: SET: SPI flag set RESET: SPI flag not set

10.2.9 duet_spi_get_interrupt_status

Function	Get SPI interrupt status
Parameters	● SPIx: SPI interface addr ● spi_interrupt: SPI interrupt flag
Return	Result: SET: corresponding SPI interrupt occurred RESET: no corresponding SPI interrupt occurred

10.2.10 duet_spi_interrupt_clear

Function	Clear SPI interrupt status
Parameters	● SPIx: SPI interface addr ● spi_interrupt: SPI interrupt flag
Return	N/A

10.2.11 duet_spi_send

Function	SPI device sends the specified sized data
Parameters	● spi: the device to send data ● data: buffer to be sent ● size: data size ● timeout: timeout in millisecond, not use, reserved for future
Return	Result: 0: success EIO: if an error occurs in any step

10.2.12 duet_spi_finalize

Function	De-initialize an SPI interface, and stop the SPI function.
Parameters	● spi: the device for which the SPI port should be de-initialized
Return	Result: 0: success EIO: if an error occurs in any step

11. ADC

11.1 Functional Description

The ADC has eight external channels which can measure the interface voltage of corresponding channel and two internal channels (used to sample temperature). The reference voltage for the ADC of ASR582X is 1.2 V, with the resolution of 12 bits.

Voltage Vol= 0.4243*adc_regdat+6.9805 (mV), temperature tempr= (data_n- data_p)*0.29 /5.25+41.5 (°C) (data_n: channel ADC_CHANNEL_TEMN register data, data_p: channel ADC_CHANNEL_TEMP register data).

Structure **duet_adc_dev_t** is used to describe ASR582X ADC hardware, and the definition is shown below:

Table 11-1 duet_adc_dev_t structure

Member Variable Name	Type	Description
port	uint8_t	channel index, rang: 0-7 (pad4-11) 8~9 (internal temperature channel)
config	duet_adc_config_t	ADC configuration: ● sampling_cycle: reserve parameter, reset to 0
priv	void *	user-defined private data

11.2 APIs

11.2.1 duet_adc_init

Function	Initialize an ADC interface, and set the ADC function
Parameters	● duet_adc_dev_t *adc_config
Return	Result: 0

11.2.2 duet_adc_get

Function	get interface input voltage digital value
Parameters	● duet_adc_dev_t *adc_config
Return	Result: interface input voltage digital value (uint: mV)

11.2.3 duet_tempr_get

Function	get the temperature value
Parameters	● duet_adc_dev_t *adc_config
Return	Result: interface input voltage digital value (unit: °C)

11.2.4 duet_adc_finalize

Function	De-initialize an ADC interface, stop the ADC function.
Parameters	● duet_adc_dev_t *adc_config
Return	Result: interface input voltage digital value (unit: °C)

12. EFUSE

12.1 Functional Description

The ASR582X integrates 4K-bit one-time programmable memory (eFuse), with the default value of 0. It can only be written from 0 to 1.

12.2 APIs

12.2.1 duet_efuse_init

Function	efuse init, must be called before read/write operation. if efuse write is needed, ldo25_open must be set to 1. If only efuse read is needed, then ldo25_open should be set to 0
Parameters	● ldo25_open: 1 - open ldo25 for efuse write operation
Return	void

12.2.2 duet_efuse_byte_read

Function	read one efuse byte
Parameters	● addr: efuse addr for read, from 0x000 to 0x1FF
Return	Result: Data in corresponding efuse address

12.2.3 duet_efuse_word_read

Function	read one efuse word
Parameters	● addr: efuse addr for read, from 0x000 to 0x1FC
Return	Result: Data in corresponding efuse address

12.2.4 duet_efuse_multi_read

Function	read multiple efuse Bytes
Parameters	● start_addr: efuse addr for write, from 0x000 to 0x1FC ● size_in_bytes: how many bytes to be read ● *pData: where efuse data is stored
Return	void

13. RAM Layout

13.1 Functional Description

The 224 KB TCM is divided into ITCM and DTCM. The 128 KB RAM can be configured for SoC, Wi-Fi and Bluetooth. Besides, there is 32 KB RAM for Wi-Fi and ADC, which belongs to Wi-Fi by default. So, there're several layout schemes for all of the memory partitions. It's easy to configure the RAM layout with the duet_ram_layout_init API and obtain the current RAM layout with the duet_get_ram_layout API.

13.2 APIs

13.2.1 duet_ram_layout_init

Function	Ram layout init. The left RAM space after configuration will belong to soc.
Parameters	● tcm_config: The configuration type of tcm ITCM_DTCM_32_192 ITCM_DTCM_96_128 ● wifi_config: The configuration type of wifi WIFI_RAM_0 WIFI_RAM_32 WIFI_RAM_64 WIFI_RAM_96 ● bt_config: The configuration type of Bluetooth BT_RAM_0 BT_RAM_16 BT_RAM_32
Return	Result: 0: success EIO: if an error occurs in any step
Notes	This function must be called before the using of Wi-Fi and BLE RAM

13.2.2 duet_get_ram_layout

Function	Get the current RAM layout parameters
Parameters	● ram_layout [OUT]: The layout parameters of the hole tcm and ram. Details showed in Ram_Layout_Type
Return	Result: 0: success EIO: if an error occurs in any step

14. PSRAM

14.1 Functional Description

It is possible that the current RAM is not enough for some productions, so a plug-in PSRAM chip will be used to extend the RAM using, which can work via SPI or QSPI interface. The address space of the PSRAM begins from PSRAM_AMBA_BASE, the value of which can't be changed.

14.2 APIs

14.2.1 psram_set_channel

Function	PSRAM sets the channel used
Parameters	● channel: The sets of GPIO pads used for PSRAM PSRAM_CHANNEL_4_9 PSRAM_CHANNEL_16_21
Return	Result: 0: success EIO: if an error occurs in any step
Notes	This function must be called before PSRAM config

14.2.2 psram_config

Function	PSRAM config.
Parameters	● mode: The PSRAM config mode PSRAM_MODE_SPI PSRAM_MODE_QSPI
Return	Result: 0: success EIO: if an error occurs in any step
Notes	This function must be called before the using of the PSRAM