CMOSTEK AN282

CMT216xA API Function Library User Guide

Overview

This document mainly discusses the functions and usage of the API function liabrary for the CMT216xA to help users have a quick start of the product development.

The product models covered in this document are shown in the table below.

Table 1. Product Models Covered in This Document

Product Model	Single-end PA	Differential PA	12-Bit ADC	Operational Amplifier	Low-frequency Wakeup	External 32.768 kHz	Packaging
CMT2160A		●	4-ch				SOP14
CMT2162A	●		8-ch		●		SSOP20
CMT2163A	●	●	9-ch		●	●	TSSOP28
CMT2165A	●		12-ch	●		●	TSSOP28
CMT2168A	●		12-ch	●	●	●	QFN32

Notes:

1. The performance and parameter details as well as the package size, silk screen and ordering information of each chip model are NOT covered in this document, please refer to the datasheet document of each chip model for details. For specific function details of the CMT216xA series, please refer to CMT216xA User Guide.

Reading guidelines:

1. This document is target for users who are well grounded in the CMT216xA and require to query API details during product development. If for a quick start of product development, users can refer to the *AN284 Development Environment Establishment* for details.
2. Suggest users read *CMT216xA User Guide* first to get familiar with basic product function information then read this document for more comprehensive understanding on the product.
3. This document serves as a quick API query manual during user coding development.

CMOSTEK Confidential

Table of Contents

CMOSTEK Confidential

CMOSTEK Confidential

CMOSTEK Confidential

CMOSTEK Confidential

CMOSTEK Confidential

1 General Introduction

The API function library of CMT216xA is divided into 5 major classes according to the on-chip function modules. Each class has several subsets.

1. System related API functions
   - System clock related API functions
   - System storage related API functions
   - System interrupt related API functions
   - API function related to system operating mode
   - Other subsidiary API functions

2. Timer related API functions
   - Timer1 related API functions
   - TimerA/B related API functions
   - Sleep Timer related API functions

3. Pin interface related API functions
   - LED function related API function
   - GPIO output / input related API functions
   - SPI interface related API functions
   - UART interface related API functions

4. Algorithm related API functions
   - AES-128 related API functions
   - True random number (TRNG) related API functions
   - CRC-16 related API functions

5. API function of the functional module
   - 3D low-frequency wakeup/receiving module related API functions
   - Sub-1G high-frequency transmitting module related API functions
   - AFE module related API functions
   - LBD module related API functions

The following sections will describe the APIs by classes and subsets.

CMOSTEK Confidential

2 System Related API Functions

System related functions are divided into 4 subsets.

- System clock related APIs including on-chip HXOSC, HFOSC, LPOSC, LXOSC and other clock modules related subsets.
  - HXOSC, off-chip 26 MHz crystal oscillator.
  - HFOSC, on-chip high-speed RC oscillator with selectable frequencies, 24 MHz, 12 MHz and 3 MHz.
  - LXOSC, off-chip 32.768 kHz crystal oscillator.
  - LPOSC, on-chip low-speed and low-power RC oscillator with a frequency of 32 kHz.
- System storage related: on-chip code running space, SFR register access, EEPROM access related subset
- System interrupt related subset.
- System working mode related subset.
- Other additional subset.

The related APIs are summarized in the below table.

Table 2. System Related API Function List

Function Class		Function Name	Used Stack Storage Space (Byte)
System clock related API	Function configuration	sys_set_hfosc_clk_sel	2
		sys_change_system_main_clock	2
		sys_set_system_clk_divider	2
		sys_enable_xo_crystal	0
		sys_disable_xo_crystal	0
		sys_enable_low_32khz_clock	2
		sys_disable_low_32khz_clock	2
		sys_enable_lfxo_crystal	2
	Module calibration	cal_hfosc_clk_high_coase_calibration	4
		cal_start_hfosc_clk_high_fine_calibration	2
		cal_stop_hfosc_clk_high_fine_calibration	2
		cal_lfosc_clk_coase_calibration	4
		cal_start_lfosc_clk_fine_calibration	2
		cal_stop_lfosc_clk_fine_calibration	2
System stroage related API	Memory related	mctl_set_source_addr	0
		mctl_set_destion_addr	0
		mctl_set_copy_size	0
		mctl_start_operation_and_wait_done	0
	SFR block1 area	sys_set_sfr_bank	0
		sys_get_sfr_bank	0

CMOSTEK Confidential

Function Class		Function Name	Used Stack Storage Space (Byte)
	SFR block0 area	sys_write_hv_reg	0
		sys_read_hv_reg	0
		sys_set_hv_reg	0
	EEROM area	eeprom_write_words	4
		eeprom_read_words	4
		eeprom_set_dec_count	6
		eeprom_get_dec_count	6
System interrupt related API	External interrupt configuration	sys_set_external_interrupt0_config	0
		sys_set_external_interrupt1_config	0
		sys_set_external_interrupt2_config	0
		sys_set_external_interrupt3_config	0
		sys_set_external_interrupt4_config	0
		sys_set_external_interrupt5_config	0
		sys_set_external_interrupt6_config	0
		sys_set_external_interrupt7_config	0
	Interrupt on-site protection	sys_push_bank0_r0_2_r7_to_stack	8
		sys_pop_bank0_r7_2_r0_from_stack	0
		sys_store_hv_interface	0
		sys_restore_hv_interface	0
System operating mode related API		sys_set_mcu_to_stop_state	0
		sys_set_mcu_to_idle_state	0
		sys_shutdown	2
Other additional API		sys_delay_10_cpu_clock_cycle	0
		sys_delay_times_cpu_clks_cycle	2
		sys_delay_us_count	0
		sys_is_first_power_up	2
		sys_clear_system_flag	0
		sys_get_product_id	0
		sys_get_uuid	0

CMOSTEK Confidential

2.1 System Clock Related API Function

The followings are system clock configuration related APIs.

2.1.1 sys_set_hfosc_clk_sel

Declaration: void sys_set_hfosc_clk_sel(uint8_t hfosc_clk_sel)

Function: set/select the clock frequency of the hfosc clock, supporting 24 MHz/12 MHz/3 MHs。

Input parameter description:

- hfosc_clk_sel: the hfoscclock clock frequency selection, supporting 3 options.
    - HFOSC_CLK_SEL_24MHZ, indicating the selection of the internal 24 MHz RC clock.
    - HFOSC_CLK_SEL_12MHZ, indicating the selection of the internal 12 MHz RC clock.
    - HFOSC_CLK_SEL_3MHZ, indicating the selection of the internal 3 MHz RC clock.

Output parameter description: none

Notes:

1. If users do not need to modify the clock frequency, it's not necessary to call this function in the user program. The system selects the 24 MHz RC clock by default in this case.
2. The macros, HFOSC_CLK_SEL_24MHz, HFOSC_CLK_SEL_12MHz, and HFOSC_CLK_SEL_3MHz, are defined in the file cmt216xa_macro.h.
3. The system operating current consumption decreases as the main frequency decreases.
4. When calling this function to perform HFOSC frequency switching, it is recommended to perform a coarse calibration process after switching through calling the function cal_hfosc_clk_high_coase_calibration (See Section 2.1.9 for details).

2.1.2 sys_change_system_main_clock

Declaration: void sys_change_system_main_clock(uint8_t clk_sw)

Function: select the main system clock, namely the HXOSC crystal clock or HFOSC clock.

Input parameter description:

- clk_sw: clock source selection

  1: select the HXOSC clock, which is an external 26 MHz crystal oscillator.

  0: select the HFOSC clock, which is an internal high-speed RC oscillator with 3 options, 24 MHz, 12 MHz and 3MHz.

Output parameter description: none

Notes:

1. User programs do not need to call this function if switching clock source is not needed. The system selects the internal 24 MHz RC clock by default in the case.

CMOSTEK Confidential

2. When users select the external HXOSC crystal, the system clock frequency is the 26 MHz frequency divided by 2, namely the 13 MHz main frequency clock.

3. When users select the external HXOSC crystal as the system clock source, it needs to ensure the XO crystal oscillator circuit is enabled before switching, namely needs to call the sys_enable_xo_crystal function (See Section 2.1.4 for details of sys_enable_xo_crystal).

4. When user programs enter the Shut Down (referred to as SDN) mode, the system will automatically call sys_disable_xo_crystal to power down the HXOSC module for power saving. However, it should be noted that when users enable the external HXOSC crystal, the system will select the 3 MHz speed internal RC oscillator (HFOSC) forcedly. Therefore the system will operate with HFOSC at 3 MHz speed first upon the next system wakeup, until the user program runs the function sys_change_system_main_clock, which switches to the XO crystal clock, then it operates according to the user's required operating speed. Thus, if users have specific requirements on operating speed upon wakeup before clock switching, it is recommended to switch the HFOSC to 24 MHz by calling the sys_set_hfosc_clk_sel function (See Section 2.1.1 for details) before entering the SDN, thus the system runs high-speed clock upon the next wakeup.

2.1.3 sys_set_system_clk_divider

Declaration: void sys_set_system_clk_divider(uint8_t clk_sys_div)

Function: set the frequency division factor of the system clock.

Input parameter description:

- clk_sys_div: the selection of the system clock frequency division factor.

Decimal Value	Binary Value	Frequency Division Factor
0～7	4'b0000~4'b0111	1
8	4'b1000	2
9	4'b1001	4
10	4'b1010	8
11	4'b1011	16
12	4'b1100	32
13	4'b1101	64
14	4'b1110	128
15	4'b1111	256

Output parameter description: none

2.1.4 sys_enable_xo_crystal

Declaration: void sys_enable_xo_crystal(void)

Function: enable the external HXOSC crystal (26 MHz) and wait for its stabilization.

Input parameter description: none

CMOSTEK Confidential

Output parameter description: none

Notes:

1. When the HXOSC crystal is enabled, the current consumption of the HXOSC crystal oscillator will increase. The typical value is 400 uA.
2. The execution time of this function mainly depends on the length of the crystal start-up, which is related to the material of the crystal itself. The typical value is 350 us according to the measurement of CMT2168A-EM.

2.1.5 sys_disable_xo_crystal

Declaration: void sys_disable_xo_crystal(void)

Function: disable the HXOSC crystal

Input parameter description: none.

Output parameter description: none.

2.1.6 sys_enable_low_32khz_clock

Declaration: void sys_enable_low_32khz_clock(uint8_t source)

Function: enable the internal low-frequency clock module (LFOSC module) and select the low-frequency clock source.

Input parameter description:

- source: the low-frequency clock source selection

  0: select the internal low-power 32 kHz low-frequency RC oscillator, LPOSC.

  1: select the external low-frequency 32.768 kHz crystal oscillator, LXOSC (also known as LFXO). In this case, the A0 and D0 pins need to be connected with a 32.768 kHz crystal.

Output parameter description: none

2.1.7 sys_disable_low_32khz_clock

Declaration: void sys_disable_low_32khz_clock(void)

Function: disable the internal low-frequency clock module, namely stop the LFOSC module (stop it no matter the LFOSC adopts LPOSC or LXOSC).

Input parameter description: none

Output parameter description: none

CMOSTEK Confidential

2.1.8 sys_enable_lfxo_crystal

Declaration: void sys_enable_lfxo_crystal(uint8_t lfxo_ibias_code)

Function: enable the external 32.768 kHz low-frequency crystal clock module, namely the LFOSC module selects LXOSC. The A0 and D0 pins need to be connected with the 32.768 kHz crystal in the case.

Input parameter description:

- lfxo_ibias_code: 32 KHZ low-frequency crystal bias current selection (0~31, the larger the value, the faster the low-frequency crystal starts-up, and the more power it consumes during startup).

Notes:

The below items are applicable to Section 2.1.6~2.1.8, in which the LFOSC module is discussed.

1. The LFOSC clock source can select either the internal RC (internal LPOSC) or the external crystal (namely LXOSC), however it does not suppport selecting both the 2 options.
2. Call sys_enable_low_32khz_clock(0) if it needs to use LPOSC.
3. It is recommended to call sys_enable_low_32khz_clock(1) when needs to use LXOSC. Since calling sys_enable_low_32khz_clock(1) already contains the excution of sys_enable_lfxo_crystal(15), there it's no need to call sys_enable_lfxo_crystal.
4. When users needs to use LXOSC and speed up LXOSC startup, call sys_enable_lfxo_crystal first, then call sys_enable_low_32khz_clock.
5. When users need to stop the LFOSC module (whether LPOSC or LXOSC is selected), call sys_disable_low_32khz_clock directly.

The followings are system clock calibration related API functions.

2.1.9 cal_hfosc_clk_high_coase_calibration

Declaration: void cal_hfosc_clk_high_coase_calibration(void)

Function: perform the rough calibration of the high-frequency RC oscillation frequency (HFOSC).

Input parameter description: none

Output parameter description: none

Notes:

1. The function performs single calibration process currently, which costs a typical time of 680 us.
2. Upon first chip power-on, the calibration function is called automatically by the system internally. If there is a critical requirment on operation accuracy of the HFOSC and the environment (temperature) changes frequently for a long time, it is recommended in the user program to design rough callibration regularly or at appropriate time through calling this function.
3. Through configuring HFOSC_CLKOUT_EN, the HFOSC clock can be output to a specified pin for analysis.

CMOSTEK Confidential

2.1.10cal_start_hfosc_clk_high_fine_calibration

Declaration: void cal_start_hfosc_clk_high_fine_calibration(void)

Function: perform the fine calibration of the high-frequency RC oscillation frequency (HFOSC). Once it starts, HFOSC will continue to make the fine-calibration until the function cal_stop_hfosc_clk_high_fine_calibration (see Section 2.1.11 for details) is called.

Input parameter description: none

Output parameter description: none

Notes:

1. The function continues monitoring the callibration of the system HFOSC to make sure HFOSC is callibrated in time during system operating. Once called, the function will keep taking effect until the stop function (see Section 2.1.11 for details) is called.
2. If this fine callibration is enabled, be sure to call the stop function to turn off this fine callibration before entering the SDN mode.
3. If there exists sudden change in temperature in the user product application environment,meanwhile, the HFOSC oscillation accuracy should be ensured, it's recommended for users to turn on the fine callibration function.

2.1.11 cal_stop_hfosc_clk_high_fine_calibration

Declaration: void cal_stop_hfosc_clk_high_fine_calibration(void)

Function: stop the internal high-frequency HFOSC clock fine calibration, cooperating with the function cal_start_hfosc_clk_high_fine_calibration (see Section 2.1.10 for details) .

Input parameter description: none

Output parameter description: none

2.1.12 cal_lfosc_clk_coase_calibration

Declaration: void cal_lfosc_clk_coase_calibration(void)

Function: perform the rough calibration function of the 32 khz low-frequency oscillator (LPOSC)

Input parameter description: none

Output parameter description: none

Notes:

1. The function performs single calibration process, costing a typical time of 680 us.
2. Upon first chip power-on, the calibration function is called automatically inside the system. Users can call this function to perform more rough callibrations at approproite time according to the specific application scenario requirments.

CMOSTEK Confidential

3. Before calling this callibration function, users need to call sys_enable_xo_crystal (see Section 2.1.4 for details), namely enable HXOSC as its calibration source.

4. Through configuring LFOSC_CLKOUT_EN, the LFOSC clock can be output to a specified pin for analysis.

5. This calibration function is only used when the internal LPOSC is selected for LFOSC. It's does not apply to LXOSC.

2.1.13cal_start_lfosc_clk_fine_calibration

Declaration: void cal_start_lfosc_clk_fine_calibration(void)

Function: start the fine-calibration function of the low-frequency RC oscillator (LPOSC, 32kHz). Once this function is called, the 32 khz low-frequency oscillator will continuously perform the fine-calibration until the function stopping the fine-calibration function of the 32 khz low-frequency oscillator (see Section 2.1.15, cal_stop_lfosc_clk_fine_calibration) is called.

Input parameter description: none

Output parameter description: none

Notes:

1. The function continues monitoring the callibration of the system LFOSC to make sure LPOSC *is callibrated in time* during system operating. Once called, the function will keep taking effect untile the stop function (see Section 2.1.14 for details) is called.

2. As sys_enable_xo_crystal (see Section 2.1.4) needs to be called to enable HXOSC as the LPOSC calibration source, additional current consumption is required. Therefore, it is recommended that users better use coarse calibration processing.

3. This calibration function is only used when LFOSC selects the internal LPOSC and does not apply to LXOSC.

2.1.14 cal_stop_lfosc_clk_fine_calibration

Declaration: void cal_stop_lfosc_clk_fine_calibration(void)

Function: stop the fine-calibration on the ow-frequency oscillator (LPOSC, 32kHz), cooperating with the function cal_start_lfosc_clk_fine_calibration (see Section 2.1.14 for details).

Input parameter description: none

Output parameter description: none

CMOSTEK Confidential

2.2 System Storage Related API

The followings are program space related APIs.

2.2.1 mctl_set_source_addr

Declaration: void mctl_set_source_addr(uint16_t src_addr)

Function: set the source address of the memory operation.

Input parameter description:

- src_addr：the source address of the memory operation.

Output parameter description: none

2.2.2 mctl_set_destion_addr

Declaration: void mctl_set_destion_addr(uint16_t dest_addr)

Function: set the destination address of the memory operation.

Input parameter description:

- dest_addr: the target address of the memory operation.

Output parameter description: none

2.2.3 mctl_set_copy_size

Declaration: void mctl_set_copy_size (uint16_t copy_size)

Function: set the number of bytes of the memory operation.

Input parameter description:

- copy_size: the number of bytes of the memory operation.

Output parameter description: None

2.2.4 mctl_start_operation_and_wait_done

Declaration: void mctl_start_operation_and_wait_done(uint8_t mem_cmd)

Function: startup the memory controller then wait for its completion. This function will suspend the MCU until it's completed.

Input parameter description:

- mem_cmd: the memory operation command. Only the MCTL_CMD_DMA command (refer to the definition in cmt216xa_macro.h) is supported currently, which requires the operation source address and destination address, as physical address, being passed to the address setting function.

Output parameter description: none

CMOSTEK Confidential

Notes:

1. The 4 memory control functions in Section 2.2.1~2.2.4 are related to the bootloader usage. Users can refer to the bootloader example provided by CMOSTEK to get better understanding if if need to write bootloader code. In general, the above 4 functions are not used if bootloader coding is not required.

The followings are Block 1 register area related APIs.

2.2.5 sys_set_sfr_bank

Declaration: void sys_set_sfr_bank(uint8_t bank)

Function: set the Bank for system accessing the SFR register in Block1 area. The main area Block1 for SFR register has 2 sub-areas Bank0 and Bank1. Users can choose to point the system access interface to a specified Bank area through this function.

Input parameter description:

- bank: the bank pointer through which the system accesses the SFR register in Block 1.

Value	Bank Selection
0	BANK0
1	BANK1

Output parameter description: none

CMOSTEK Confidential

Notes:

1. When accessing the registers located in the Block1 area, users must ensure the current bank pointer is correct. It is recommended that users call sys_set_sfr_bank switching to the targe bank first when accessing the registers in the Block1 area. The reason for doing this is, when users call the API function, the Bank switching will be performed as required by the API function execution process, however the Bank pointer will not be restored after the API execution.
2. The API library can cover most of the register accessing in Block 1. Therefore, to save code space, it is recommended uses better call API functions to have register accessing rather than access the registers directly, which requires an extra operation of bank switching, namely switch band through calling this function.
3. The following registers in the Block1 area can be operated irespective of the Bank pointer.

Address	SFR Name	Function
0x80	P0	Port0 register
0x81	SP	Stack pointer
0x82	DPL	DPTR pointer
0x83	DPH	
0x87	PCON	Power control register
0x88	TCON	Timer1 control reigster
0x89	TMOD	Timer1 configuration register

Address	SFR Name	Function
0x8B	TL1	Timer1 timer
0x8C	TH1	
0x98	SCON0	Series port 0 control register
0x99	SBUF0	Series port 0 data buffer register
0xA8	IEN0	Interrupt enabling register 0
0xB8	IPL0	Interrupt priority register 0
0xD0	PSW	System flag register
0xE0	ACC	Accumulator
0xE6	IEN1	Interrupt enabling register 1
0xF0	B	Register B
0xF1	IRCON1	Interrupt request flag register
0xF6	IPL1	Interrupt priority register 1

2.2.6 sys_get_sfr_bank

Declaration: uint8_t sys_get_sfr_bank(void)

Function: get the current Bank pointer for accessing the SFR in Block 1 area.

Input parameter description: none

Output parameter description:

- the Bank pointer through which the system is accessing the SFR BANK area. .

Return Value	Bank Selection
0	BANK0
1	BANK1

The followings are Block 0 area related APIs.

2.2.7 sys_write_hv_reg

Declaration: void sys_write_hv_reg(uint8_t addr, uint8_t val)

Function: the writing interface fucntion of the SFR registers in the always-on Block0 area. It writes a value to the specified address of Block0 area.

Input parameter description:

- addr: the address of SFR register of the always-on Block0 area
- val: the value to be written to the addr address.

Output parameter description: none

CMOSTEK Confidential

2.2.8 sys_read_hv_reg

Declaration: uint8_t sys_read_hv_reg(uint8_t addr):

Function: the reading interface of the always-on Block0 area, which reads the value of a specified address in Block0 area.

Input parameter description:

- addr: the address of SFR register in the always-on Block0 domain.

Output parameter description: the data value stored in the addr address

2.2.9 sys_set_hv_reg

Declaration: void sys_set_hv_reg(uint8_t addr, uint8_t val, uint8_t mask)

Function: the SFR register operating interface of the always-on domain Block0, which operates the specified bits of the specified register address.For example, when users need to write the decimal value *3* to the bit [6:4] of the address 0x05 (addr = 0x05) while keeping other bits unchanged, users only need to call this function as sys_set_hv_reg(0x05, (3 << 4), 0x70).

Input parameter description:

- addr: the SFR register address of the always-on Block0 area.
- val: the value to be written to some bits
- mask: the mask value of the operated bits, e.g. the mask value corresponding to bit[7:6] is 0xc0.

Output parameter description: none

The followings are EEPROM related APIs.

2.2.10 eeprom_write_words

Declaration: uint8_t eeprom _write_words(uint8_t start_addr, uint8_t xdata *write_buf, uint8_t eeprom_size)

Function: write data directly to a specified address of the EEPROM (including erasing and writing), operating the EEPROM in the MSB mode. This area is a 32-Word storage space (16 bits are 1 Word).The address range available for user operation is 0x00 ~ 0x1F. Users should not operate the EEPROM out of the range, otherwise it will cause unpredictable errors.

Input parameter description:

- start_addr: the starting address of the EEPROM to be written. Note that it must be less than 0x20.
- write_buf: xdata type array / variable pointer, pointing to the data address to be written to the EEPROM.
- eeprom_size: the amount of EEPROM data to be written. Note that it must satisfy start_addr + eeprom_size <= 0x20.

Output parameter description:

0: indicates that the writing operation is successful.

1: indicates that the erasing operation fails.

2: Indicates that the writing operation fails.

Notes:

1. The operating unit is a Word in EEPROM. namely 16 bits.

CMOSTEK Confidential

2.2.11 eeprom_read_words

Declaration: uint8_t eeprom_read_words(uint8_t start_addr, uint8_t xdata *read_buf, uint8_t eeprom_size)

Function: read data directly from a specified address of the EEPROM, operating the EEPROM in the MSB mode. This area is a 32-Word storage space (16 bits are 1 Word).The address range available for user operation is 0x00 ~ 0x1F. Users should not operate the EEPROM out of the range, otherwise it will cause unpredictable errors.

Input parameter description:

- start_addr: the starting address of the EEPROM to be read, note that it must be less than 0x20.
- read_buf: xdata type array/variable pointer, pointing to the data address to be read from the EEPROM;.
- eeprom_size: the amount of EEPROM data to be written. Note that it must satisfy start_addr + eeprom_size <= 0x20.

Output parameter description:

0: indicates that the reading operation is successful.

1: indicates that the reading operation fails.

Notes:

The operating unit is a Word in EEPROM. namely 16 bits.

2.2.12 eeprom_set_dec_count

Declaration: uint8_t eeprom_set_dec_count(uint8_t index, uint32_t *dec_value)

Function: Write incremental/decremental value in a specific way thus to achieve optimized writing endurance. Strictly, the on-chip EEPROM actually is a type of MTP, with its original endurance level inferior to the EEPROMs which can endure millions of operations. However, aming at writting values satisfying specific increasing or decreasing rules, the writing operation can be optimized using some algorithm to achieve the endurance of more than one million operations without damage. Writing in this way, it requires 3 EEPROM addresses in each operation (namely 16 × 3 = 48 bits data), and it requires the stored RMS value to be less than 0x3F0000 (namely 4,128,768 in decimal), near to 22 valid bits, however, it needs to occupy a 48-bit storage space.

Input parameter description:

- Index: the operation index number in a unit of 3 EEPROM storage spaces, which can only be 0, 3, 6, 9, ..., 24, 27, namely up to 10 address groups.
- Dec_value: uint32_t variable pointer pointing to the variable to be written. The valid data width is 22 bits.

Output parameter description:

0: indicates that the operation is successful.

1: indicates that the erasing operation fails.

2: indicates that the writing operation fails.

Notes:

1. This writing operation mode is different from that of eeprom_write_words (see Section 2.2.10). The eeprom_write_words is a direct operation mode while this function provides a particular writing operation for the increment or decrement value. A typical example of incremental value application is the synchronous counter applied in the synchronous counter encryption mechanism adopted in a RKE system. The incremental value is increased by 1 each time it is read from the EEPROM and then it is written to the EEPROM. Futhermore, when writing a value unsatisfying the incremental or decremental rule, or a

CMOSTEK Confidential

value without a regular pattern, to the EEPROM, it cannot fully benefit from the optimized algorithm. However, it can achieve the same writing endurance as that of eeprom_write_words at worst.

2. In user programs, both eeprom_write_words and eeprom_set_dec_count can be called althogether, however, users should pay special attention to the written address arrangement to avoid mutual influence.
3. When writing using this function, users must read using eeprom_set_dec_count (see Section 2.2.13), namely cannot use eeprom_read_words (see Section 2.2.11 for details).

2.2.13 eeprom_get_dec_count

Declaration: uint8_t eeprom_get_dec_count(uint8_t index, uint32_t *dec_value)

Function: co-working eeprom_set_dec_count, the function is to read the value written in EEPROM by eeprom_set_dec_count.

Input parameter description:

- Index: the operation index number in a unit of 3 EEPROM storage spaces, which can only be 0, 3, 6, 9, ..., 24, 27.
- Dec_value: uint32_t variable pointer to the variable where the read value will be stored.

Output parameter description:

0: indicates that the reading operation is successful.

1: indicates that the reading operation fails.

Note:

1. The reading operation mode is different from that of eeprom_read_words (see Section 2.2.11). Eeprom_read_words applies a direct reading mode while this function applies a reading mode matching with eeprom_set_dec_count.
2. In the user program, users can call both eeprom_read_words and eeprom_get_dec_count altogether, however the reading address should match with the address used in the corresponding writing function to avoid reading errors.
3. When reading with this function, the corresponding writing function should be eeprom_get_dec_count (see Section 2.2.12 for details) and should not be eeprom_write_words (see Section 2.2.10 for details).

2.3 API Functions for System Module

The followings are system clock configuration related APIs.

2.3.1 sys_set_external_interrupt0_config

Declaration: void sys_set_external_interrupt0_config(unit8_t cfg)

Function: select the event and polarity for system external interrupt 0.

Input parameter description:

- cfg: the configuration value of system external interrupt 0, corresponding to the IRQ0_SEL register (located in Bank0 of Block1).

CMOSTEK Confidential

Bit7	6	5	4	3	2	1	Bit0
Reserved	IRQ0_SW	IRQ0_SEL[5:0]					

Bit_6, IRQ0_SW: interrupt source polarity selection

Set to 1: trigger interrupt on the rising edge or at high level

Set to 0: trigger interrupt on the falling edge or at low level

Bit_5:0, IRQ0_SEL: select among 35 interrupt events

Refer to the interrupt related sections in *CMT216xA User Manual* and *CMT216xA Register Manual*, or *Section 9.1 Interrupt Events* in this document for details.

Output parameter description: None.

2.3.2 sys_set_external_interrupt1_config

Declaration: void sys_set_external_interrupt1_config(uint8_t cfg)

Function: select the event and polarity for system external interrupt 1.

Input parameter description:

- cfg: the configuration value of system external interrupt 0, corresponding to the IRQ1_SEL register (located in Bank0 of Block1).

Bit7	6	5	4	3	2	1	Bit0
Reserved	IRQ1_SW	IRQ1_SEL[5:0]					

Bit_6, IRQ1_SW: interrupt source polarity selection

Set to 1: trigger interrupt on the rising edge or at high level

Set to 0: trigger interrupt on the falling edge or at low level

Bit_5:0, IRQ1_SEL: select among 35 interrupt events

Refer to the interrupt related sections in CMT216xA User Manual and CMT216xA Register Manual, or Section 9.1 Interrupt Events in this document for details.

Output parameter description: none.

2.3.3 sys_set_external_interrupt2_config

Declaration: void sys_set_external_interrupt2_config(uint8_t cfg)

Function: select the event and polarity for system external interrupt 2.

Input parameter description:

- cfg: the configuration value of system external interrupt 2, corresponding to the IRQ2_SEL register (located in Bank0 of Block1).

CMOSTEK Confidential

Bit7	6	5	4	3	2	1	Bit0
Reserved	IRQ2_SW	IRQ2_SEL[5:0]					

Bit_6, IRQ2_SW: interrupt source polarity selection

Set to 1: trigger interrupt on the rising edge or at high level

Set to 0: trigger interrupt on the falling edge or at low level

Bit_5:0, IRQ2_SEL: select among 35 interrupt events

Refer to the interrupt related sections in CMT216xA User Manual and CMT216xA Register Manual, or Section 9.1 Interrupt Events in this document for details.

Output parameter description: none.

2.3.4 sys_set_external_interrupt3_config

Declaration: void sys_set_external_interrupt3_config(uint8_t cfg)

Function: set the event and polarity selection for system external interrupt 3.

Input parameter description:

- cfg: the configuration value of system external interrupt 3, corresponding to the IRQ3_SEL register (located in Bank0 of Block1). The parameter description is the same as that of IRQ2, which is omitted here.

Output parameter description: none.

2.3.5 sys_set_external_interrupt4_config

Declaration: void sys_set_external_interrupt4_config(uint8_t cfg)

Function: set the event and polarity selection for system external interrupt 4.

Input parameter description:

- cfg: the configuration value of system external interrupt 4, corresponding to the IRQ4_SEL register (located in Bank0 of Block1). The option values and description are same as those of IRQ2, which are imitted here.

Output parameter description: none.

2.3.6 sys_set_external_interrupt5_config

Declaration: void sys_set_external_interrupt5_config(uint8_t cfg)

Function: set the event and polarity selection for system external interrupt 5.

Input parameter description:

- cfg: the configuration value of system external interrupt 5, corresponding to the IRQ5_SEL register (located in Bank0 of Block1). The option values and description are same as those of IRQ2, which are imitted here.

Output parameter description: none.

CMOSTEK Confidential

2.3.7 sys_set_external_interrupt6_config

Declaration: void sys_set_external_interrupt6_config(uint8_t cfg)

Function: set the event and polarity selection for system external interrupt 6.

Input parameter description:

- cfg: the configuration value of system external interrupt 6, corresponding to the IRQ6_SEL register (located in Bank0 of Block1). The option values and description are same as those of IRQ2, which are imitted here.

Output parameter description: none.

2.3.8 sys_set_external_interrupt7_config

Declaration: void sys_set_external_interrupt7_config(uint8_t cfg)

Function: set the event and polarity selection for system external interrupt 7.

Input parameter description:

- cfg: the configuration value of system external interrupt 7, corresponding to the IRQ7_SEL register (located in Bank0 of Block1). The option values and description are same as those of IRQ2, which are imitted here.

Output parameter description: none.

2.3.9 sys_push_bank0_r0_2_r7_to_stack

Declaration: void sys_push_bank0_r0_2_r7_to_stack(void)

Function: push R0~R7 (Bank0) into stack. This function is called in interrupt service routines.

Input parameter description: none

Output parameter description: none.

2.3.10 sys_pop_bank0_r7_2_r0_from_stack

Declaration: void sys_pop_bank0_r7_2_r0_from_stack(void)

Function: pop R0~R7(bank0) from the stack, which is called in interrupt service routines.

Input parameter description: none

Output parameter description: none

CMOSTEK Confidential

2.3.11 sys_store_hv_interface

Declaration: uint8_t sys_store_hv_interface(void)

Function: get the register interface accessing state of the always-on domain (namely Block0). This function is called in interrupt service routines to save register interface accessing state of the always-on domain when the interrupt is triggered, then to restore the state at the end of the interrupt service routine.

Input parameter description: none

Output parameter description: the register interface accessing state of the always-on domain

2.3.12 sys_restore_hv_interface

Declaration: void sys_restore_hv_interface(uint8_t hv_store)

Function: restore the register interface accessing state of the always-on domain.

Input parameter description:

- hv_store: the register interface accessing state of the always-on domain.

Output parameter description: none

2.4 System Operating Mode Related API

2.4.1 sys_set_mcu_to_stop_state

Declaration: void sys_set_mcu_to_stop_state(void)

Function: set the 8051 to enter the STOP state. The $F_{PCLK\ and}$ and F_{MCLK} clocks are stopped in this state.

Input parameter description: none

Output parameter description: none

Notes:

1. In the STOP mode, the 8051 core can only be woken up by external interrupt 0 and external interrupt 1.
2. The STOP mode stops the 8051 core only, the other modules in the main area are still in operating, therefore the chip is not in the lowest power status. The lowest power mode of the chip is the SDN mode, which can be entered by calling sys_shutdown (see Section 2.4.3 for details).

2.4.2 sys_set_mcu_to_idle_state

Declaration: void sys_set_mcu_to_idle_state(void)

Function: set the 8051 to enter the idle state with the F_{MCLK} clock stopped while the F_{PCLK} clock still in operating.

Input parameter description: none

CMOSTEK Confidential

Output parameter description: none

Notes:

1. In the IDLE mode, any interrupt can wake up the 8051 core.
2. The IDLE mode means the 8051 is idle, but the other modules in the main area are still in operating, therefore the chip is not in the lowest power status. The lowest power mode of the chip is the SDN mode, which can be entered by calling sys_shutdown (see Section 2.4.3 for details).

2.4.3 sys_shutdown

Declaration: void sys_shutdown(void)

Function: power down the main area of the system with the function modules in the always-on area operating in a low-power state.

Input parameter description: none

Output parameter description: none

Notes:

1. The SDN mode is the lowest power consumption mode for this chip series.
2. In the SDN mode, since the main area is completely powered down, the operating storage space such as PRAM and XRAM are all cleared due to power loss. For the variables that users will continue using (namely continue processing the varibles after the next wakeup), they need to be written to the reserved space in the always-in Block0 area before it enters the SDN mode. For the writing interface function, see Section 2.2.7~2.2.9 for details. For the storage space allowed to be written, please refer to *CMT216xA User Manual* and *CMT216xA Register Manual* for details.
3. The main area is powered down, all the Block1 area (Bank0 and Bank1) registers are powered down, and the involved function modules are disabled in the SDN mode.
4. In the SDN mode, upon the next wakeup of the chip, users should follow the principle of wakeup first, then interrupt, that is, the code is loaded first after wakeup, then it's possible to execute an interrupt service routine only after the code starts running.

CMOSTEK Confidential

2.5 Other Additional APIs

2.5.1 sys_delay_10_cpu_clock_cycle

Declaration: void sys_delay_10_cpu_clock_cycle(void)

Function: software time delay function. It waits for 10 F_{MCLK} clock cycles.

Input parameter description: none

Output parameter description: none

2.5.2 sys_delay_times_cpu_clks_cycle

Declaration: void sys_delay_times_cpu_clks_cycle(uint16_t times_clks)

Function: software time delay function. It waits for 30*times_clks F_{MCLK} clock cycles.

Input parameter description:

- times_clks: the number F_{MCLK} clock cycles to delay. Unit: 30* F_{MCLK} timing unit

Output parameter description: none

2.5.3 sys_delay_us_count

Declaration: void sys_delay_us_count(uint8_t count)

Function: hardware time delay function. the delay time is in us, up to 255 us.

Input parameter description:

- count: the number of delay time in us

Output parameter description: none

Notes:

1. This function is the hardware delay function. Although the timing process is not affected by F_{MCLK}, entering call, exiting call, jump and configuration are still affected. Therefore when users switch or lower the operating frequencies, or in the case of requiring measurement accuracy per 1 time unit, it may exist errors. By default, users can use the 24 MHz internal RC to consider the typical value.

2.5.4 sys_is_first_power_up

Declaration: uint8_t sys_is_first_power_up(void)

Function: detect whether the system is powered on for the first time, which is one of the most frequently called functions for the user program to judge the first power-on then to perform the corresponding initial configuration.

Input parameter description: none

Output parameter description:

1: the first power-on

0: not the first power-on

Notes:

1. When entering the SDN mode, upon the next wakeup, the user program need to be reloaded and run. Therefore this function is used to identify whether it is a first power-on or a wakeup.

CMOSTEK Confidential

2.5.5 sys_clear_system_flag

Declaration: void sys_clear_system_flag(void)

Function: initialize and clear some important system flags, which is one of the most frequently called functions in the program initialization stage.

Input parameter description: none

Output parameter description: none

2.5.6 sys_get_product_id

Declaration: uint16_t sys_get_product_id(void)

Function: get the 16-bit product ID (chip ID)

Input parameter description: none

Output parameter description: chip ID

2.5.7 sys_get_uuid

Declaration: void sys_get_uuid(uint8_t xdata *uuid)

Function: get the on-chip 32-bit UUID

Input parameter description:

- *uuid: xdata type variable pointer pointing to the cache address of the 32-bit UUID.

Output parameter description: none

CMOSTEK Confidential

3 Timer Related API

The timer related functions are divided into 3 subsets.

- Timer1 related API functions
- TimerA/B related API functions
- Sleep timer related API functions

The related APIs are listed in the below table.

Table 3. Timer Related API

Function Category		Function Name	Used Stack Space (Byte)
Timer1 related API function	Interface mapping configuration	sys_set_timer1_input_from_gpio	0
Timer A / B related API function	Interface mapping configuration	sys_set_timer_a_cci0_input_from_gpio	0
		sys_set_timer_a_cci1_input_from_gpio	0
		sys_set_timer_b_cci0_input_from_gpio	0
		sys_set_timer_b_cci1_input_from_gpio	0
Sleep timer related API function	Function configuration	sys_enable_sleep_timer	2
		sys_disable_sleep_timer	2
		sys_set_sleep_timer_working_mode	2
		sys_set_sleep_timer_m_r_value	2

3.1 Timer1 Related API

3.1.1 sys_set_timer1_input_from_gpio

Declaration: void sys_set_timer1_input_from_gpio(uint8_t gpio_num)

Function: select the Timer1's capture input channel from GPIO0~GPIO15.

Input parameter description:

- gpio_num: data input channel selection. See Section 9.8 *Function Module Input Selection GPIO list* for details

Output parameter description: none

CMOSTEK Confidential

3.2 Timer A / B Related API

3.2.1 sys_set_timer_a_cci0_input_from_gpio

Declaration: void sys_set_timer_a_cci0_input_from_gpio(uint8_t gpio_num)

Function: select the CCI0 input channel of TimerA from GPIO0～GPIO15.

Input parameter description:

- gpio_num: TA_CCI0 input channel selection. See Section 9.8 *Function Module Input Selection GPIO list* for details

Output parameter description: none

3.2.2 sys_set_timer_a_cci1_input_from_gpio

Declaration: void sys_set_timer_a_cci1_input_from_gpio(uint8_t gpio_num)

Function: select the CCI1 input channel of TimerA from GPIO0～GPIO15.

Input parameter description:

- gpio_num: TA_CCI1 input channel selection. See Section 9.8 *Function Module Input Selection GPIO list* for details

Output parameter description: none

3.2.3 sys_set_timer_b_cci0_input_from_gpio

Declaration: void sys_set_timer_b_cci0_input_from_gpio(uint8_t gpio_num)

Function: select the CCI0 input channel of TimerB from GPIO0～GPIO15.

Input parameter description:

- gpio_num: TB_CCI0 input channel selection. See Section 9.8 *Function Module Input Selection GPIO list* for details

Output parameter description: none

3.2.4 sys_set_timer_b_cci1_input_from_gpio

Declaration: void sys_set_timer_b_cci1_input_from_gpio(uint8_t gpio_num)

Function: select the CCI1 input channel of TimerB from GPIO0～GPIO15.

Input parameter description:

- gpio_num: TB_CCI1 input channel selection. See Section 9.8 *Function Module Input Selection GPIO list* for

CMOSTEK Confidential

details

Output parameter description: none

3.3 Sleep Timer Related API

3.3.1 sys_enable_sleep_timer

Declaration: void sys_enable_sleep_timer(void)

Function: enable the sleep timer.

Input parameter description: none

Output parameter description: none

3.3.2 sys_disable_sleep_timer

Declaration: void sys_disable_sleep_timer(void)

Function: disable the sleep timer.

Input parameter description: none

Output parameter description: none

3.3.3 sys_set_sleep_timer_working_mode

Declaration: void sys_set_sleep_timer_working_mode(uint8_t working_mode)

Function: set the opreating mode of the sleep timer. The sleep timer supports 2 operating modes currently, RTC mode and WAKEUP mode.

Input parameter description:

- working_mode: the operating mode of the timer:
  SLEEP_TIMER_RTC_MODE: RTC mode.
  SLEEP_TIMER_WAKEUP_MODE: WAKEUP mode.

Output parameter description: none

Notes:

1. SLEEP_TIMER_RTC_MODE and SLEEP_TIMER_WAKEUP_MODE are defined in the cmt216xa_macro.h file.
2. Please refer to *CMT216A User Guide* for the difference between RTC mode and WAKEUP mode.

CMOSTEK Confidential

3.3.4 sys_set_sleep_timer_m_r_value

Declaration: void sys_set_sleep_timer_m_r_value(uint8_t m_value, uint8_t r_value)

Function: set the overflow time of the sleep timer. Overflow timer = m[11:0] * 2^(r[3:0] +1) * 31.25 us.

$$T_{\text{SLEEP\_TIMER}} = \mathrm{m} \times 2^{(\mathrm{r}+1)} \times 31.25us$$

Input parameter description:

- m_value: the lower 8 bits of the m value, namely m[7:0].

Bit7	6	5	4	3	2	1	Bit0
m[7:0]							

- r_value: the higher 4 bits of the m value and r value.

Bit7	6	5	4	3	2	1	Bit0
m[11:8]				r[3:0]			

Output parameter description: none

CMOSTEK Confidential

4 GPIO Interface Related API

The pin interface related API functions are divided into the following 4 subsets.

- LED function related API function
- GPIO output/input related API functions
- SPI interface related API functions
- UART interface related API functions

The function is summarized in the below table.

Table 4. Pin Interface Related API Function List

Function		Function Name	Used Stack Space（Byte）
LED function related API	Interface mapping configuration	sys_enable_led_flash	2
GPIO output/input related API	GPIO output configuration	sys_set_gpio0_out_sel	0
		sys_set_gpio1_out_sel	0
		sys_set_gpio2_out_sel	0
		sys_set_gpio3_out_sel	0
		sys_set_gpio4_out_sel	0
		sys_set_gpio5_out_sel	0
		sys_set_gpio6_out_sel	0
		sys_set_gpio7_out_sel	0
		sys_set_gpio8_out_sel	0
		sys_set_gpio9_out_sel	0
		sys_set_gpio10_out_sel	0
		sys_set_gpio11_out_sel	0
		sys_set_gpio12_out_sel	0
		sys_set_gpio13_out_sel	0
		sys_set_gpio14_out_sel	0
		sys_set_gpio15_out_sel	0
	GPIO output/input control	sys_set_gpio_hold	2
		sys_get_input_value_from_gpio0_7	0
		sys_get_input_value_from_gpio8_15	0
		sys_set_output_value_to_gpio0_7	0
		sys_set_output_value_to_gpio8_15	0
		sys_get_output_value_from_gpio0_7	0
		sys_get_output_value_from_gpio8_15	0
	Port 0 input mapping configuration	sys_set_p0_0_input_from_gpio	0
		sys_set_p0_1_input_from_gpio	0

CMOSTEK Confidential

Function		Function Name	Used Stack Space（Byte）
		sys_set_p0_2_input_from_gpio	0
		sys_set_p0_3_input_from_gpio	0
		sys_set_p0_4_input_from_gpio	0
		sys_set_p0_5_input_from_gpio	0
		sys_set_p0_6_input_from_gpio	0
		sys_set_p0_7_input_from_gpio	0
SPI interface related API	Interface mapping configuration	sys_set_spi_nss_input_from_gpio	0
		sys_set_spi_sck_input_from_gpio	0
		sys_set_spi_miso_input_from_gpio	0
		sys_set_spi_mosi_input_from_gpio	0
	Function configuration	spi_setup_config	0
		spi_set_one_line_direction	0
		spi_set_enabled	0
		spi_set_software_nss	0
UART interface related API	Interface mapping configuration	sys_set_uart_rxd_input_from_gpio	0

4.1 LED Function Related API

4.1.1 sys_enable_led_flash

Declaration: void sys_enable_led_flash(uint8_t led_cfg)

Function: LED module function configuration

Input parameter description:

- Led_cfg：the LED module configuration value corresponding to the LED_CTL register (at the address of 0xCC in Bank0 of Block1)

  Bit_7: reserved.

  Bit_6: LED module enabling bit. 0: disable, 1: enable.

  Bit_5: signal source of LED output. 0: TX data, 1: PWM signal.

  Bit_4: PWM output rate. 0: 3.3 kHz, 1: 6.68 kHz.

  Bit_3:0: duty cycle selection of PWM with a configuration value range of 0 ~ 15, namely there're 16 duty cycle options increasing by a step of 1/16.

Output parameter description: none

CMOSTEK Confidential

4.2 GPIO Output/Input Related APIs

The followings are GPIO output mapping related APIs.

4.2.1 sys_set_gpio0_out_sel

Declaration: void sys_set_gpio0_out_sel(uint8_t gpio_out_sel)

Function: select the signal source of the GPIO0 output corresponding to the A0 pin.

Input parameter description:

- gpio_out_sel: the signal source of GPIO0. Refer to Section 9.7 *GPIO Output Mapping List* for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[0]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[0]	Mapto P0.0 of 8051, supporting bit operation mode
2	4'b0010	tb_out0	Timer B capture compare output 0
3	4'b0011	tb_out1	Timer B capture compare output 1
4	4'b0100	tb_out2	Timer B capture compare output 2
5	4'b0101	nss_out	Chip selection signal pin for SPI (master mode)
6	4'b0110	sck_out	Clock signal pin for SPI (master mode)
7	4'b0111	miso_out	MISO signal pin for SPI (slave mode)
8	4'b1000	mosi_out	MOSI signal pin for SPI (master mode)
9	4'b1001	rxd0_out	RXD output pin of 8051 serial port 0 (synchronous serial mode)
10	4'b1010	txd0	TxD output pin of 8051 serial port
11	4'b1011	ta_out0	Timer A capture compare output 0
12	4'b1100	ta_out1	Timer A capture compare output 1
13	4'b1101	ta_out2	Timer A capture compare output 2
14	4'b1110	led_out_lv	LED function output pin
15	4'b1111	t1_ov	8051 Timer1 overflow signal pin

Output parameter description: none

CMOSTEK confidential

4.2.2 sys_set_gpio1_out_sel

Declaration: void sys_set_gpio1_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO1, corresponding to the A1 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO1. Refer to 9.7 *GPIO Output Mapping List* for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[1]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[1]	Map to P0.1 of 8051, supporting bit operation mode
2~15	Others	Same as the corresponding options of GPIO0	

Output parameter description: none

4.2.3 sys_set_gpio2_out_sel

Declaration: void sys_set_gpio2_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO2, corresponding to the A2 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO2. Refer to 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[2]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[2]	Map to P0.2 of 8051, supporting bit operation mode
2~15	Others	Same as the corresponding options of GPIO0	

Output parameter description: none

CMOSTEK Confidential

4.2.4 sys_set_gpio3_out_sel

Declaration: void sys_set_gpio3_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO3, corresponding to the A3 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO3. Refer to 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[3]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[3]	Map to P0.3 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

4.2.5 sys_set_gpio4_out_sel

Declaration: void sys_set_gpio4_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO4 output, corresponding to the A4 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO4. Refer to 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[4]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[4]	Map to P0.4 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

4.2.6 sys_set_gpio5_out_sel

Declaration: void sys_set_gpio5_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO5, corresponding to the A5 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO5. Refer to Section 9.7 *GPIO Output Mapping List* or the

CMOSTEK Confidential

below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[5]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[5]	Map to P0.5 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

4.2.7 sys_set_gpio6_out_sel

Declaration: void sys_set_gpio6_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO6, corresponding to the A6 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO6. Refer to Section 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[6]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[6]	Map to P0.6 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

4.2.8 sys_set_gpio7_out_sel

Declaration: void sys_set_gpio7_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO7 output, corresponding to the A7 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO7. Refer to Section 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[7]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[7]	Map to P0.7 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

CMOSTEK Confidential

Output parameter description: none

4.2.9 sys_set_gpio8_out_sel

Declaration: void sys_set_gpio8_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO8, corresponding to the B0 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO8. Refer to 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[7]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[7]	Map to P0.0 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

4.2.10 sys_set_gpio9_out_sel

Declaration: void sys_set_gpio9_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO9, corresponding to the B1 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO9. Refer to 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[9]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[1]	Map P0.1 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

CMOSTEK Confidential

4.2.11 sys_set_gpio10_out_sel

Declaration: void sys_set_gpio10_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO10, corresponding to the B2 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO10. Refer to Section 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[10]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[2]	Map to P0.2 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

4.2.12 sys_set_gpio11_out_sel

Declaration: void sys_set_gpio11_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO11, corresponding to the B3 pin.

Input parameter description:

- gpio_out_sel: the signal source of GPIO11. Refer to Section 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[11]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[3]	Map to P0.3 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

4.2.13 sys_set_gpio12_out_sel

Declaration: void sys_set_gpio12_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO12, corresponding to the B4 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO12. Refer to Section 9.7 *GPIO Output Mapping List* or the below table for details.

CMOSTEK Confidential

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[12]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[4]	Map to P0.4 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

4.2.14 sys_set_gpio13_out_sel

Declaration: void sys_set_gpio13_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO13, corresponding to the B5 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO13. Refer to Section 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[13]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[5]	Map to P0.5 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

4.2.15 sys_set_gpio14_out_sel

Declaration: void sys_set_gpio14_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO14, corresponding to the B6 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO14. Refer to Section 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[14]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[6]	Map to P0.6 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

CMOSTEK Confidential

4.2.16 sys_set_gpio15_out_sel

Declaration: void sys_set_gpio15_out_sel(uint8_t gpio_out_sel)

Function: select the output signal source of GPIO15, corresponding to the B7 pin.

Input parameter description:

- gpio_out_sel: the output signal source selection of GPIO15. Refer to Section 9.7 *GPIO Output Mapping List* or the below table for details.

Decimal Value	Binary Value	GPIO Function	Description
0	4'b0000	gpio_out_r[15]	Controlled by GPIO_OUT_R register (by default)
1	4'b0001	p0[7]	Map to P0.7 of 8051, supporting bit operation mode
2~15	Others	Same as corresponding options of GPIO0	

Output parameter description: none

The followings are GPIO input/output control related APIs.

4.2.17 sys_set_gpio_hold

Declaration: void sys_set_gpio_hold(uint8_t gpio_hold)

Function: set the output data source of all GPIO

Input parameter description:

- gpio_hold: the selection of gpio output data source
  GPIO_HOLD_ENA: select from the GPIOx_ODR register in the always-on domain .
  GPIO_HOLD_DIS: select from the GPIOx_OUT_SEL function selection and configuration in the miain area.

Output parameter description: none

Notes:

1. sys_set_gpio_hold is mainly used to maintain the level state of the output port in the SDN mode. In the SDN mode, as the main area is in the power-down state, thus the GPIO selection function configured by the user program (by configuring GPIOx_OUT_SEL) is no longer valid. Therefore, before entering the SDN mode, it is necessary to call sys_set_gpio_hold(GPIO_HOLD_ENA) to process the level state of the output port through the always-on GPIOx_ODR register.
2. If the option GPIO_HOLD_ENA is selected when the system is in the operating mode (like being woken up) with the main area in operating, it needs to control the port output status by the register GPIOx_ODR. However, as this register is in the always-on domain, it requires to call pass sys_write_hv_reg or sys_set_hv_reg to operate it (see 2.2.7 and 2.2.9), which is inconvenient if the program is used frequently. Therefore, it is recommended to select the option GPIO_HOLD_DIS when the system is in the operating mode while select GPIO_HOLD_ENA in the SDN mode.
3. As this function does not apply to input function ports, users can also consider switching the output port to the input mode

CMOSTEK Confidential

and setting the pull-up state (matching with the external device/function of the chip) before entering the SDN mode. In this case it's not necessary to call this function.

4. The two macros GPIO_HOLD_DIS and GPIO_HOLD_ENA are defined in the cmt216xa_macro.h file.

4.2.18 sys_get_input_value_from_gpio0_7

Declaration: uint8_t sys_get_input_value_from_gpio0_7(void)

Function: read the input value of GPIO0~GPIO7, that is, access the register GPIO_IN_R_L Bank0 in Block1 area, 0xC1).

Input parameter description: none

Output parameter description: the input value of GPIO0~7

Bit7	6	5	4	3	2	1	Bit0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

4.2.19 sys_get_input_value_from_gpio8_15

Declaration: uint8_t sys_get_input_value_from_gpio 8_15 (void)

Function: read the input value of GPIO8~GPIO15, that is, access the register GPIO_IN_R_H (Bank0 in Block1 area, 0xC0).

Input parameter description: none

Output parameter description: the input value of GPIO8~15

Bit7	6	5	4	3	2	1	Bit0
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8

4.2.20 sys_set_output_value_to_gpio0_7

Declaration: void sys_set_output_value_to_gpio0_7(uint8_t gpio_value)

Function: set the GPIO0~GPIO7 output value. The function is valid only if the GPIO signal source is selected as GPIO_OUT_R (namely the default selection of gpio_out_sel, the value 0, see Section 4.2.1~4.2.8 for details). The effect is the same as that of writting to GPIO_OUT_R_L (0xC3 in Bank0 of Block1 area).

Input parameter description: none

Output parameter description: the input value of GPIO0~7

Bit7	6	5	4	3	2	1	Bit0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

CMOSTEK Confidential

4.2.21 sys_set_output_value_to_gpio8_15

Declaration: void sys_set_output_value_to_gpio8_15(uint8_t gpio_value)

Function: set the GPIO8~GPIO15 output value. The function is valid only if the GPIO signal source is selected as GPIO_OUT_R (namely the default selection of gpio_out_sel, the value 0, see Section 4.2.9～4.2.16 for details). The effect is the same as that of writting to GPIO_OUT_R_H (0xC2 in Bank0 of Block1 area).

Input parameter description:

- gpio_value: GPIO output value

Bit7	6	5	4	3	2	1	Bit0
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8

Output parameter description: none

4.2.22 sys_get_output_value_from_gpio0_7

Declaration: uint8_t sys_get_output_value_from_gpio0_7(void)

Function: read the output value of GPIO0~7. This function is valid only if the signal source of GPIO is selected as GPIO_OUT_R. The effect is the same as that of reading GPIO_OUT_R_L (0xC3 in Bank0 of Block1 area).

Input parameter description: none

Output parameter description: the output value of GPIO0~GPIO7

Bit7	6	5	4	3	2	1	Bit0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

4.2.23 sys_get_output_value_from_gpio8_15

Declaration: uint8_t sys_get_output_value_from_gpio8_15(void)

Function: read the output value of GPIO8~GPIO15. This function is valid only if the signal source of GPIO is selected as GPIO_OUT_R. The effect is the same as that of reading GPIO_OUT_R_H (0xC2 in Bank0 of Block1 area).

Input parameter description: none

Output parameter description: the output value of GPIO

Bit7	6	5	4	3	2	1	Bit0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

CMOSTEK Confidential

The followings are port0 inputmmapping selection related APIs.

4.2.24 sys_set_p0_0_input_from_gpio

Declaration: void sys_set_p0_0_input_from_gpio(uint8_t gpio_num)

Function: select the data input channel of the 8051's P0.0.

Input parameter description:

- gpio_num: data input channel selection of P0.0. See Section 9.8 *Function Module Input Selection GPIO list* or the below table for details.

  The Port0 input channel selection list is shown in the below table.

Table 5. Port0 Input Channel Selection List

Decimal Value	Channel Selection	Pin
0	GPIO0	A0
1	GPIO1	A1
2	GPIO2	A2
…	…	…
6	GPIO6	A6
7	GPIO7	A7
8	GPIO8	B0
9	GPIO9	B1
…	…	…
14	GPIO14	B6
15	GPIO15	B7
Others	Invalid	Invalid

Output parameter description: none

4.2.25 sys_set_p0_1_input_from_gpio

Declaration: void sys_set_p0_1_input_from_gpio(uint8_t gpio_num)

Function: select the data input channel of the 8051's P0.1.

Input parameter description:

- gpio_num: data input channel selection of P0.1. See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

Output parameter description: none

CMOSTEK Confidential

4.2.26 sys_set_p0_2_input_from_gpio

Declaration: void sys_set_p0_2_input_from_gpio(uint8_t gpio_num)

Function: select the data input channel of the 8051's P0.2.

Input parameter description:

- gpio_num: data input channel selection of P0.2. See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

Output parameter description: none

4.2.27 sys_set_p0_3_input_from_gpio

Declaration: void sys_set_p0_3_input_from_gpio(uint8_t gpio_num)

Function: select the data input channel of the 8051's P0.3.

Input parameter description:

- gpio_num: data input channel selection of P0.3. See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

Output parameter description: none

4.2.28 sys_set_p0_4_input_from_gpio

Declaration: void sys_set_p0_4_input_from_gpio(uint8_t gpio_num)

Function: select the data input channel of the 8051's P0.4.

Input parameter description:

- gpio_num: data input channel selection of P0.4. See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

Output parameter description: none

4.2.29 sys_set_p0_5_input_from_gpio

Declaration: void sys_set_p0_5_input_from_gpio(uint8_t gpio_num)

Function: select the data input channel of the 8051's P0.5.

Input parameter description:

- gpio_num: data input channel selection of P0.5. See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

Output parameter description: none

CMOSTEK Confidential

4.2.30 sys_set_p0_6_input_from_gpio

Declaration: void sys_set_p0_6_input_from_gpio(uint8_t gpio_num)

Function: select the data input channel of the 8051's P0.6.

Input parameter description:

- gpio_num: data input channel selection of P0.6. See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

Output parameter description: none

4.2.31 sys_set_p0_7_input_from_gpio

Declaration: void sys_set_p0_7_input_from_gpio(uint8_t gpio_num)

Function: select the data input channel of the 8051's P0.7.

Input parameter description:

- gpio_num: data input channel selection of P0.7. See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

Output parameter description: none

4.3 SPI Interface Related API

The followings are SPI mapping selection related APIs.

4.3.1 sys_set_spi_nss_input_from_gpio

Declaration: void sys_set_spi_nss_input_from_gpio(uint8_t gpio_num)

Function: select the NSS input channel of the SPI bus (slave mode)

Input parameter description:

- gpio_num: the NSS input channel selection of the SPI bus (slave mode). See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

Output parameter description: none

4.3.2 sys_set_spi_sck_input_from_gpio

Declaration: void sys_set_spi_sck_input_from_gpio(uint8_t gpio_num)

Function: select the SCK input channel of the SPI bus (slave mode)

Input parameter description:

- gpio_num: the SCK input channel selection of the SPI bus (slave mode). See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

CMOSTEK Confidential

Output parameter description: none

4.3.3 sys_set_spi_miso_input_from_gpio

Declaration: void sys_set_spi_miso_input_from_gpio(uint8_t gpio_num)

Function: select the MISO input channel of the SPI bus (master mode)

Input parameter description:

- gpio_num: the MISO input channel selection of the SPI bus (master mode). See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

Output parameter description: none

4.3.4 sys_set_spi_mosi_input_from_gpio

Declaration: void sys_set_spi_mosi_input_from_gpio(uint8_t gpio_num)

Function: select the MISO input channel of the SPI bus (slave mode)

Input parameter description:

- gpio_num: the MISO input channel selection of the SPI bus (slave mode). See Section 9.8 *Function Module Input Selection GPIO list* or Table 5 for details.

Output parameter description: none

The followings are SPI function configuration related APIs.

4.3.5 spi_setup_config

Declaration: void spi_setup_config(STRU_SPI_SETUP xdata *ptr_cfg)

Function: configure the SPI bus parameters.

Input parameter description:

- ptr_cfg: the STRU_SPI_SETUP structure pointer, pointing to SPI configuration parameters. See Section 9.5 *SPI Interface Configuration Structure* for details.

Output parameter description: none

4.3.6 spi_set_one_line_direction

Declaration: void spi_set_one_line_direction(uint8_t dir)

Function: set the transmission direction of the SPI single-wire bidirectional mode

Input parameter description:

- dir: data line transmission direction.

CMOSTEK Confidential

0, the data line is used as the input mode, namely the Rx. mode.

1, the data line is used as the output mode, namely the Tx. mode.

Output parameter description: none

4.3.7 spi_set_enabled

Declaration: void spi_set_enabled(uint8_t enable)

Function: enable/disable the SPI bus .

Input parameter description:

- enable

  0, disable the SPI bus.

  1, enable the SPI bus.

Output parameter description: none

4.3.8 spi_set_software_nss

Declaration: void spi_set_software_nss(uint8_t level)

Function: set the level of the SPI NSS software mode (SSM=1, see the STRU_SPI_SETUP structure definition for details).

Input parameter description:

- level:

  0 = NSS pin output low level.

  1 = NSS pin output high level.

Output parameter description: none

4.4 UART Input Mapping Selection API

4.4.1 sys_set_uart_rxd_input_from_gpio

Declaration: void sys_set_uart_rxd_input_from_gpio(uint8_t gpio_num)

Function: select the RxD data input channel of the UART

Input parameter description:

- gpio_num: the UART RxD data input channel selection. See Section 9.8 *Function Module Input Selection GPIO* list or Table 5 for details.

Output parameter description: none

CMOSTEK Confidential

5 Algorithm Related API

Algorithm-related API functions are divided into the below 3 subsets.

- AES-128 related API functions
- True random number (TRNG) related API functions
- CRC-16 related API functions

The information is summarized in the below table.

Table 6. Algorithm Related API List

Function Category	Function Name	Used Stack Space（Byte）
CRC-16 function	sys_crc16_calc	2
True random number (TRNG) generation function	sys_get_random_data	0
AES-128 calculation related function	aes_generate_decrypted_key	2
	aes_encryption	2
	aes_decryption	2

5.1 CRC-16 Calculation API

5.1.1 sys_crc16_calc

Declaration: void uint16_t sys_crc16_calc(STRU_CRC16_CFG *cfg)

Function: 16-bit CRC configuration

Input parameter description:

- cfg: the STRU_CRC16_CFG structure pointer, pointing to the CRC-16 calculation configuration. See Appendix *9.6 CRC-16 Structure Description* for details.

Output parameter description: return the CRC16 calculation result

CMOSTEK Confidential

5.2 True Random Number Generation API

5.2.1 sys_get_random_data

Declaration: void sys_get_random_data(uint8_t xdata *random_data, uint8_t byte_cnt)

Function: generate a random number of bytes_cnt bytes (in a unit of byte).

Input parameter description:

- random_data: xdata type array/variable pointer pointing to the address where the random number is stored.
- byte_cnt: the number of bytes of the random number to be generated.

Output parameter description: none

Notes:

1. As the true random number is generated by calculation based on the frequency difference between the two clock sources, LFOSC and HFOSC_REF, it is necessary to ensure that the 2 modules are enabled before calling this function, as shown in the below program example.

```
#define M_PD_HFOSC_REF 0x40                                    //HFOSC_REF control macro definition
Sys_enable_low_32khz_clock(0);                                 //open the LFOSC module
Sys_set_hv_reg(CUS_AFE8_ADDR, 0, M_PD_HFOSC_REF);              //open the HFOSC_REF module
Sys_get_random_data(g_random_number, 16);                      //call true random function
Sys_set_hv_reg(CUS_AFE8_ADDR, M_PD_HFOSC_REF, M_PD_HFOSC_REF); //turn off the HFOSC_REF module
```

In above,as the HFOSC_REF module is irrelated to other functions, users should close it use to save power; Furthermore, if the LFOSC is not involved in other functions in the program, it can also be closed by calling the sys_disable_low_32khz_clock function to save power.

5.3 AES-128 Calculation Related API

5.3.1 aes_generate_decrypted_key

Declaration: void aes_generate_decrypted_key(uint8_t idata *ptr_encrypt_key)

Function: generate the AES decryption key using the original AES encryption key. It is called to generate the decryption key before calling the AES decryption function.

Input parameter description:

- ptr_encrypt_key: an array of idata type pointing to the address of the 16-byte AES original key. The decryption key is stored in the address space that the pointer points to, after the function call completes.

Output parameter description: none

CMOSTEK Confidential

5.3.2 aes_encryption

Declaration: void aes_encryption(uint8_t idata *ptr_aes_data,uint8_t idata *ptr_encrypt_key)

Function: perform AES-128 encryption on the 16-byte plaintext.

Input parameter description:

- ptr_aes_data: an idata type array pointer pointing to the first address of a 16-byte plaintext array. After the function exection completes, the generated 16-byte ciphertext is stored in the address space.
- ptr_encrypt_key: an idata type array pointing to the first address of a 16-byte key array. After the function call completes, the original key in stored the address space will be destroyed and cannot be used for the next encryption.

Output parameter description: none

Notes:

1. Since the pointed key array is destroyed after the operation, it is necessary to assign the key value again when it is necessary to use the same key to do the operation again.

5.3.3 aes_decryption

Declaration: void aes_decryption(uint8_t idata *ptr_decrypt_data, uint8_t idata *ptr_decrypt_key)

Function: perform AES-128 decryption to the 16-byte AES-128 ciphertext.

Input parameter description:

- ptr_decrypt_data: an idata type array pointer pointing to the first address of a 16-byte AES-128 ciphertext array. After the function execution completes, the decrypted 16-byte plaintext is stored in the address space.
- ptr_decrypt_key: am idata type array pointer pointing to the first address of a 16-byte decryption key array generated by the aes_generate_decrypted_key function. After the function execution completes, the key of stored in the address space will be destroyed and cannot be used for the next decryption.

Output parameter description: none

Notes:

1. Since the decryption key array that the pointer points to is destroyed after the operation, when it is necessary to use the same key to do operation again, it is recommended to call aes_generate_decrypted_key to calculate the decryption key and cache the decryption key to another space, and then assign the decryption key again in the operation to save computing time.

CMOSTEK Confidential

6 Function Module Related API

Functional module related API functions are divided into the following 4 subsets.

- 3D low-frequency wakeup/receiving module related API functions.
- Sub-1G high-frequency transmission module related API functions.
- AFE module related API functions
- LBD module related API functions

The information is summarized in the below table,

Table 7. Function Module Related API List

Function Category	Function Name	Used Stack Space（Byte）
3D low-frequency wakeup/receiving module related API	sys_get_lfrx_state	2
	lfrx_setup_config	2
	lfrx_enable_lfrx_osc_out	2
	lfrx_disable_lfrx_osc_out	2
	cal_lfrx_osc_amplitude_calibration	8
	cal_lfrx_osc_frequency_calibration	8
Sub-1G high-frequency transmission module related API	sys_disable_pll_module	0
	sys_get_differential_pa_calibration_capacitor_code	0
	tx_modu_setup_config	0
	tx_sym_setup_config	0
	tx_sym_transmit	2
	tx_sym_prepare_for_transmission	10
	tx_sym_start_direct_transmit	0
	tx_sym_stop_direct_transmit	0
AFE module related API	sys_enable_afe_snooze	2
	sys_disable_afe_snooze	2
	sys_set_afe_sensor_channel	2
	afe_front_setup_config	2
	afe_snooze_setup_config	2
	afe_open_analog_front_power	2
	afe_close_analog_front	2
	afe_open_pir_front_power	2
	afe_close_pir_front_power	2
	afe_read_sar_adc_data	2
	afe_check_system_voltage	2
LBD module related API	sys_start_stop_supply_voltage_supervision	2
	sys_read_result_from_supply_voltage_supervision	0

CMOSTEK Confidential

6.1 3D low-frequency Wakeup/receiving Module Related API

6.1.1 sys_get_lfrx_state

Declaration: uint8_t sys_get_lfrx_state(void)

Function: get the state of the low-frequency Rx module.

Input parameter description: none

Output parameter description: the state of the low-frequency Rx module.

Decimal Value	State	Description
0	Idle	Idle mode
2	Listen	Listening mode
3	Sleep	Sleeping mode
9	Decode	Decode mode

6.1.2 lfrx_setup_config

Declaration: void lfrx_setup_config(STRU_LFRX_SETUP xdata *ptr_lfrx_setup)

Function: set the configuration parameters of the Rx module.

Input parameter description:

- ptr_lfrx_setup: the STRU_LFRX_SETUP structure pointer, pointing to the configuration parameters for low-frequency Rx module, refer to Appendix 5.3 for details.

Output parameter description: none

6.1.3 lfrx_enable_lfrx_osc_out

Declaration: void lfrx_enable_lfrx_osc_out(void)

Function: enable the low-frequency antenna oscillator output.

Input parameter description: none

Output parameter description: none

6.1.4 lfrx_disable_lfrx_osc_out

Declaration: void lfrx_disable_lfrx_osc_out(void)

Function: disable the low-frequency antenna oscillator output.

Input parameter description: none

CMOSTEK Confidential

Output parameter description: none

6.1.5 cal_lfrx_osc_amplitude_calibration

Declaration: void cal_lfrx_osc_amplitude_calibration(uint8_t lfrx_ant_ref)

Function: perform the calibration of the amplitude of the low-frequency antenna oscillator .

Input parameter description:

- Lfrx_ant_re: the reference value of the low-frequency antenna

Output parameter description: none

6.1.6 cal_lfrx_osc_frequency_calibration

Declaration: void cal_lfrx_osc_frequency_calibration(uint8_t lfrx_ant_ref)

Function: perform the frequency calibration of the of the low-frequency antenna oscillator .

Input parameter description:

- Lfrx_ant_re: the reference value of the low-frequency antenna

Output parameter description: none

6.2 Tx Module Related API

6.2.1 sys_disable_pll_module

Declaration: void sys_disable_pll_module(void)

Function: close the phase-locked loop (PLL) module.

Input parameter description: none

Output parameter description: none

6.2.2 sys_get_differential_pa_calibration_capacitor_code

Declaration: uint8_t sys_get_differential_pa_calibration_capacitor_code(uint8_t pa_power_code, uint8_t pa_idac_code)

Function: get the differential PA calibration capacitor value. Applicable when the transmitter module is configured as a differential PA output. The capacitance value obtained by this function can be used as a measure of the current differential matching performance. Generally, the capacitor value between 80 and 120 indicates a fine matching. If the capacitor value is low, such as 20, it indicates that the peripheral differential PA matching is poor, and the matching parameters need to be replaced.

Input parameter description

CMOSTEK Confidential

- pa_power_code: pa power code, take the transmission power of 10 dBm as example, the recommended value is 64.
- pa_idac_code: pa idac code, take the transmission power of 10 dBm as example, the recommended value is 20.

Output parameter description: differential PA calibration capacitor value

6.2.3 tx_modu_setup_config

Declaration: void tx_modu_setup_config(STRU_TX_MODU_SETUP xdata *ptr_stru_tx_modu_setup)

Function: set the configuration parameters of the transmission module, such as frequency point, data rate, adjustment mode, etc.

Input parameter description:

- ptr_stru_tx_modu_setup: the STRU_TX_MODU_SETUP structure pointer, pointing to the configuration parameters, refer to Appendix 9.2 for details.

Output parameter description: none

6.2.4 tx_sym_setup_config

Declaration: void tx_sym_setup_config(uint8_t tx_sym_setup)

Function: set the configuration of the transmitting interface.

Input parameter description:

- tx_sym_setup: transmission encoding configuration.
  - Bit_7:6: reserved.
  - Bit_5:3: the bit width of the transmitting buffer value.
    If set to 8, take every byte of the transmission buffer data to transmit.
    If set to 4, take only 4 bits of each byte of the transmission buffer data to transmit.
    If set to 1, take only 1 bit of each byte of the transmission buffer data to transmit.
    The direction for fetching data from the transmission buffer is determined by the Bit2 of tx_sym_setup.

Decimal value	Binary Value	Number of Transmitted Bits
0	3'b000	1
1	3'b001	2
2	3'b010	3
3	3'b011	4
4	3'b100	5
5	3'b101	6
6	3'b110	7
7	3'b111	8

CMOSTEK Confidential

- Bit_2: whether the MSB or LSB is transmitted first, namely, the buffer fetching direction.

  When set to 0, take LSB first, that is, the lower bit out first, and the data shifts to t the right.

  When set to 1, take MSB first, that is, the higher bit out first, and the data shifts to the left.

- Bit_1:0: the Tx status control after finishing buffer data transmission

Binary Value	Status Control
2'b00	Close Tx module
2'b01	Continue transmitting the last remained data in the buffer space
2'b10	Continue transmitting 0
2'b11	Continue transmitting 1

Output parameter description: none

6.2.5 tx_sym_transmit

Declaration: void tx_sym_transmit(uint8_t xdata *ptr_trans_buf,uint8_t tx_length)

Function: transmit the data to be sent. The Tx interface is closed after the transmission.

Input parameter description:

- ptr_trans_buf: a xdata type array pointer pointing to the first address of the array to be transmitted.
- tx_length: the length of the array to be transmitted.

Output parameter description: none

Notes:

1. The Tx interface has only one byte of buffer space. It is recommended that users temporarily disable the interrupt function before calling this function to avoid interrupts blocking the data transmission. If the interrupt service routine takes less time and the transmiting data rate is low, it may be not necessary to disable the interrupt function.
2. Before calling the transmitting function, the below steps should be taken.
   a) Call tx_modu_setup_config and tx_sym_setup_config to configure the transmiting mode and necessary parameters.
   b) Call tx_sym_prepare_for_transmission to ensure that the transmitting module can start normally;
   c) Call tx_sym_transmit for data transmission.
   d) According to the function and power requirements, call sys_disable_pll_module and sys_disable_xo_crystal to close the PLL and HXOSC modules to save current consumption.
3. After the function is executed, the Tx interface will be closed directly, that is, the transmission will stop.

CMOSTEK Confidential

6.2.6 tx_sym_prepare_for_transmission

Declaration: uint8_t tx_sym_prepare_for_transmission(uint8_t voltage_th)

Function: start the internal link of the transmitting module and return the related state. It includes 1) starting HXOSC crystal, VCO calibration and AFC calibration; 2) detect whether the system voltage is suitable, namely above the threshold (enough to afford the transmission consumption) and whether the phase-locked loop (PLL) is in locked status, and so on. Users can decide whether to continue data transmission according to the returned status. Notably, this function adds an additional virtual load for measurement, so that users can determine whether the current power supply can support the transmission properly.

Input parameter description:

- Voltage_th: the threshold voltage set by users, conforming to the following calculation formula.

$$\text{Vlotage\_th} = \text{Round}\left(\frac{V_{BAT}}{4.8} \times 255\right)$$

In above, V_{BAT} is the chip supply voltage.

Output parameter description: the status of the transmitting link.

- Bit_7: 2: reserved
- Bit_1: phase-locked loop (PLL) status.
  0, indicating the PLL is out of lock, and cannot continue to transmit at this time.
  1, indicating the PLL is locked, and the transmission can continue at this time;
- Bit_0: system voltage status.
  0, indicating that the detection is higher than the set threshold.
  1, indicating that the detection is lower than the set threshold.

Notes:

1. During the function execution, the battery voltage is detected and compared with the set threshold. During this voltage detection, an analog load is turned on inside the system as a load-on measurement to allow users to determine accurately whether it is appropriate to continue transmitting.
2. This function will modify the AFE detection channel for supply voltage measuring. Therefore, if there are other analogs to be sampled at the same time in user system, users need to reconfigure the AFE detection channel after calling this function or before sampling other analogs .

6.2.7 tx_sym_check_system_voltage

Declaration: uint8_t tx_sym_check_system_voltage(uint8_t voltage_th)

Function: detect whether the system voltage is higher than the threshold. Note that during the voltage detection, an additional virtual load is added for measurement.

Input parameter description:

- Voltage_th: the threshold voltage set by users, conforming to the following calculation formula.

CMOSTEK Confidential

$$\text{vlotage\_th} = \text{Round}\left(\frac{V_{BAT}}{4.8} \times 255\right)$$

In above, V_{BAT} is the chip supply voltage.

Output parameter description: system voltage status.

0, indicating that the detection is higher than the set threshold.

1, indicating that the detection is lower than the set threshold.

6.2.8 tx_sym_start_direct_transmit

Declaration: void tx_sym_start_direct_transmit(void)

Function: start the direct mode. After the direct mode is started, the software transmits by setting Bit0 of the TX_SYM_GROUP register (0xA9 address in Bank1 of Block1 area), with the transmission rate and encoding controlled by the software.

Input parameter description: none

Output parameter description: none

6.2.9 tx_sym_stop_direct_transmit

Declaration: void tx_sym_stop_direct_transmit(void)

Function: stop the direct mode, cooperating with the function tx_sym_start_direct_transmit.

Input parameter description: : none

Output parameter description: none

6.3 AFE Module Related API

6.3.1 sys_enable_afe_snooze

Declaration: void sys_enable_afe_snooze(void)

Function: enable the analog front end (AFE) SNOOZE mode.

Input parameter description: none

Output parameter description: none

Notes:

1. Before enabling the AFE SNOOZE mode, it needs to start the LFOSC module, see Section 2.1.6～2.1.8 for details.

CMOSTEK Confidential

6.3.2 sys_disable_afe_snooze

Declaration: void sys_disable_afe_snooze(void)

Function: disable the analog front end (AFE) SNOOZE mode.

Input parameter description: none

Output parameter description: none

6.3.3 sys_set_afe_sensor_channel

Declaration: void sys_set_afe_sensor_channel(uint8_t chnn)

Function: configure the specific sensor channel for the analog front end (AFE) module,which consists of HSOA1 and HSOA2. Refer to *AN281 CMT216xA AFE User Guide* for details.

Input parameter description:

- chnn: the selection of the sensor channel

Decimal Value	Binary Value	Sesor Channel Selection	Pin Occupation
0	3’b000	Close sensor measurement channel (default)	None
1	3'b001	Exernal sensor channel 1	A1、A2、ASP、ASN
2	3'b010	Exernal sensor channel 2	A3、A4、PSP、PSN
3	3'b011	Internal temperature sensor	None
Other values	Other values	Reserved	

Output parameter description: none

Notes:

1. The internal temperature sensor is not calibrated in manufacture. It can be used for reference only.

6.3.4 afe_front_setup_config

Declaration: void afe_front_setup_config(STRU_AFE_FRONT_SETUP xdata *ptr_afe_front_setup)

Function: configure the front end parameters of the AFE module. Refer to Section 9.4.1 *AFE Configuration Structure* and *AN281 CMT216xA ADC and AFE User Guide* for details.

Input parameter description:

- ptr_afe_front_setup: the STRU_AFE_FRONT_SETUP structure pointer, pointing to the front end parameter configurations of the AFE module, refer to Section 9.4.1 *AFE Configuration Structure* for details.

Output parameter description: none

CMOSTEK Confidential

6.3.5 afe_snooze_setup_config

Declaration: void afe_snooze_setup_config(STRU_AFE_SNOOZE_SETUP xdata *ptr_afe_snooze_setup)

Function: configure the AFE SNOOZE parameters. Refer to Section 9.4.1 *AFE Configuration Structure* and *AN281 CMT216xA ADC and AFE User Guide* for details.

Input parameter description:

- ptr_afe_snooze_setup: the STRU_AFE_SNOOZE_SETUP structure pointer, pointing to the AFE SNOOZE configuration parameter, refer to Section 9.4.1 *AFE Configuration Structure* for details.

Output parameter description: none

6.3.6 afe_open_analog_front_power

Declaration: void afe_open_analog_front_power(STRU_AFE_FRONT_POWERUP_SETUP xdata *ptr_front_powerup_cfg)

Function: the switch-on/off control of the AFE sub-modules. Refer to Section 9.4.1 AFE Configuration Structure and *AN281 CMT216xA ADC and AFE User Guide* for details.

Input parameter description:

- ptr_front_powerup_cfg: the STRU_AFE_FRONT_POWERUP_SETUP structure pointer, pointing to the AFE power module configuration. Refer to Section 9.4.3 *AFE Module Enabling Confogiration Structure* for details.

Output parameter description: none

6.3.7 afe_close_analog_front

Declaration: void afe_close_analog_front(void)

Function: close all analog front end (AFE) channel links and sub-modules .

Input parameter description: none

Output parameter description: none

6.3.8 afe_open_pir_front_power

Declaration: void afe_open_pir_front_power(void)

Function: enable the cascading PIR sensor detection circuit (consisting of LPOA0 and LPOA1), and related modules. Refer to *AN281 CMT216xA ADC and AFE User Guide* for details.

Input parameter description: none

Output parameter description: none

CMOSTEK Confidential

6.3.9 afe_close_pir_front_power

Declaration: void afe_close_pir_front_power(void)

Function: disable the cascading PIR sensor detection circuit (consisting of LPOA0 and LPOA1), and related modules. Refer to Refer to *AN281 CMT216xA ADC and AFE User Guide* for details.

Input parameter description: none

Output parameter description: none

6.3.10 afe_read_sar_adc_data

Declaration: uint16_t afe_read_sar_adc_data(void)

Function: read the conversion value of the SAR-ADC

Input parameter description: none

Output parameter description: the conversion value of the 12-bit SAR-ADC.

6.3.11 afe_check_system_voltage

Declaration: uint8_t afe_check_system_voltage(uint8_t voltage_th)

Function: check whether the system voltage is higher than the set threshold. Note that during the voltage measuring, no additional virtual load is added for measurement.

Input parameter description:

- voltage_th: the voltage threshold set by users. conforming to the below formula.

$$\text{vlotage\_th} = \text{Round}\left(\frac{V_{BAT}}{4.8} \times 255\right)$$

Output parameter description: whether the system voltage is higher than the threshold.
0: detected value is higher than threshold.
1: detected value is lower than threshold.

Notes:

1. The difference between afe_check_system_voltage and tx_sym_prepare_for_transmission (see 6.2.6) is that the former does not add an additional analog load measurement voltage.
2. This function will modify the AFE detection channel for supply voltage measuring. Therefore, if there are other analogs to be sampled at the same time in user system, users need to reconfigure the AFE detection channel after calling this function or before before sampling other analogs .

CMOSTEK Confidential

6.4 LBD Module Related API

6.4.1 sys_start_stop_supply_voltage_supervision

Declaration: void sys_start_stop_supply_voltage_supervision(uint8_t cfg)

Function: start/stop the power voltage monitoring function. When the power supply voltage monitoring function is started, the voltage monitoring module continuously monitors the power supply voltage of the chip. If the voltage is lower than the threshold, the monitoring result is saved, and corresponding processing is performed according to the conditions configured by users.

Input parameter description:

- cfg: monitoring condition configuration

Bit7	6	5	4	3	2	1	Bit0
Reserved					lvd_sel	pwr_rst	lvd_en

- Bit_7:3, reserved
- Bit_2, lvd_sel, voltage monitoring comparing threshold.
  Set to 0, select a threshold of 1.8 V.
  Set to 1, select a threshold of 2.0 V.
- Bit_1, pwr_rst, power reset setting, that is, reset when the power supply voltage is lower than the monitoring threshold.
  Set to 0, no reset.
  Set to 1, reset.
- Bit_0, lvd_en, open/close the power supply voltage monitoring function on or off:
  Set to 0, close.
  Set to 1, open.

Output parameter description: none.

6.4.2 sys_read_result_from_supply_voltage_supervision

Declaration: uint8_t sys_read_result_from_supply_voltage_supervision(void)

Function: read the supply voltage monitoring result, and clear the result then.

Input parameter description: none

Output parameter description:

0: supply voltage is constantly higher than the threshold voltage.

1: supply voltage is once lower than the threshold voltage. (lower than the threshold voltage at some time possibly)

CMOSTEK Confidential

7 Stack Space Related API

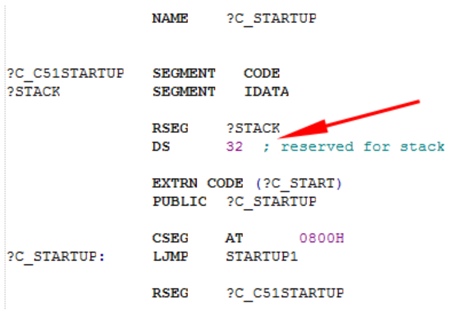
In program coding, users need to know the stack space used by the called API functions, thus to reserve the stack space in advance to prevent stack overflow errors. The stack space size used by the related API functions are listed in the function summary table at the beginning of Section 2 ~ 6. The actual stack overhead of calling an API function is the internally used stack space overhead plus the extra 2-byte stack overhead of calling the API function itself.

The maximum stack space overhead depends on whether the application uses 1 or 2 priority levels of interrupt service routines. At worst, the maximum stack overhead is that an application uses 2 priority levels of interrupts and the lower priority interrupt service routine is interrupted by a higher priority interrupt service routine. Users can estimate the maximum stack space overhead based on the API calls and the interrupt service routines used in the application, and reserve enough stack space preventing stack overflow. The methods for roughly estimating the maximum stack space overhead are listed in the below table.

Table 8. Rough Estimate Method of Maximum Stack Space

Application Program Using Single Level Priority Interrupt	Application Program Using 2-level Priority Interrupt
Maximum stack space overhead:	Maximum stack space overhead:
2 bytes for calling API function	2 bytes for calling API function
+ 10 bytes for the worst-case stack overhead of API calling	+ 10 bytes for the worst-case stack overhead of API calling
+ 2 bytes forinterrupt returning address	+ 2 bytes for interrupt service routine calling API function
+ 6 bytes for PC, DP, A, etc. protection	+ 6 bytes for PC, DP, A, etc. protection
+ 2 bytes for interrupt service routine calling API function	+ 6 bytes for the worst stack overhead called API function
+ 6 bytes for the worst stack overhead called API function	+ 6 bytes for API function
+ 8 bytes for r0~r7 protection	+ 8 bytes for r0~r7 protection
+ 4 bytes reserved	+ 2 bytes for intrrupt returning address
= 40 bytes	+ 6 bytes for PC, DP, A, etc. protection
	+ 2 bytes for interrupt service routine calling API function
	+ 6 bytes for the worst-case stack overhead of the called API
	+ 8 bytes for r0~r7 protection
	+ 4 bytes reserved
	= 64 bytes

Users can modify reserved stack space size in the file STARTUP.A51 as shown in the below figure.



```
                    NAME     ?C_STARTUP

?C_C51STARTUP       SEGMENT   CODE
?STACK              SEGMENT   IDATA

                    RSEG     ?STACK
                    DS       32   ; reserved for stack

                    EXTRN CODE (?C_START)
                    PUBLIC   ?C_STARTUP

                    CSEG     AT       0800H
?C_STARTUP:         LJMP     STARTUP1

                    RSEG     ?C_C51STARTUP
```

CMOSTEK Confidential

8 APIs Cannot be Called in Interrupt Routine

Some API functions are not allowed to be called in the interrupt service routine since they share some memory space (such as DATA, IDATA, XDATA) with other API functions. If they are called by the interrupt service routine, it may cause multiple API functions accessing the same memory space out of order, thus resulting in unpredictable consequences. The specifical API functions that cannot be called in interrupt routine are listed in the below table.

Table 9. APIs Cannot be Called in Interrupt Routine

Function Module		API Function Name
System related API	System clock related API	cal_hfosc_clk_high_coase_calibration
		cal_start_hfosc_clk_high_fine_calibration
	System storage related API	eeprom_write_words
		eeprom _read_words
		eeprom _set_dec_count
		eeprom _get_dec_count
	Other additional API	sys_delay_us_count
Pin interface relared API	SPI interface related API	spi_setup_config
Algorithm related API	CRC-16 related API	sys_crc16_calc
	AES-128 related API	aes_generate_decrypted_key
		aes_encryption
		aes_decryption
Function module related API	3D low-frequency wakeup/receiving related API	lfrx_setup_config
		cal_lfrx_osc_amplitude_calibration
		cal_lfrx_osc_frequency_calibration
	Tx module related API	sys_get_differential_pa_calibration_capacitor_code
		tx_modu_setup_config
		tx_sym_transmit
		tx_sym_prepare_for_transmission
		tx_sym_check_system_voltage
	Analog front end module related API	afe_front_setup_config
		afe_snooze_setup_config
		afe_open_analog_front_power
		afe_check_system_voltage

CMOSTEK Confidential

9 Appendix

9.1 Interrupt Event

Table 10. 35 Interrupt Events

Function Module	Decimal Value	Binary Value	Interrupt Source	Triggering Type	Description
GPIO	0~15	6'b000000~ 6'b001111	gpio_in0~ gpio_in15	Configurable	IOC interrupt event of GPIO1-15
Sleep timer	16	6'b010000	sleep_timeup	Rising edge	Sleep timer overflow interrupt event
3D low-frequency wakeup	17	6'b010001	lfrx_mcu_rclk	Rising edge	Low-frequency clock synchronization interrupt event
	18	6'b010010	lfrx_mcu_rdata	Rising edge	Low-frequency data synchronization clock interrupt event
GPIO	19	6'b010011	gpio_in16	Configurable	IOC interrupt event of GPIO16
3D low-frequency wakeup	20	6'b010100	wkid_pass	Rising edge	low-frequency WakeupID matching passed interrupt event
	21	6'b010101	sync_pass	Rising edge	low-frequency SyncWord matching passed interrupt event
Analog front end module	22	6'b010110	sar_data_update	Rising edge	SAR-ADC conversion data update interrupt event
3D low-frequency wakeup	23	6'b010111	lfrx_signal_ok	Rising edge	Low-frequency CW mode signal valid interrupt event
Timer A	24	6'b011000	tmra_ifg	Rising edge	Timer A overflow interrupt event
	25	6'b011001	taccr0_ifg	Configurable	Timer A capture input 0 interrupt event
	26	6'b011010	taccr1_ifg	Configurable	Timer A capture input 1 interrupt event
	27	6'b011011	taccr2_ifg	Configurable	Timer A capture input 2 interrupt event
Tx module	28	6'b011100	tx_sym_empty	Rising edge	Tx module transmission buffer empty interrupt event
SPI bus	29	6'b011101	spi_txe	Rising edge	SPI transmitting empty interrupt event
	30	6'b011110	spi_rxne	Rising edge	SPI receiving not empty interrupt event
Timer B	31	6'b011111	tmrb_ifg	Rising edge	Timer B overflow interrupt event
	32	6'b100000	tbccr0_ifg	Configurable	Timer B capture input 0 interrupt event
	33	6'b100001	tbccr1_ifg	Configurable	Timer B capture input 1 interrupt event
	34	6'b100010	tbccr2_ifg	Configurable	Timer B capture input 2 interrupt event

CMOSTEK Confidential

9.2 Tx Module Configuration Structure

Tx module configuration structure (STRU_TX_MODU_SETUP) is defined in file CMT216xA.H as follows.

```
typedef struct TX_MODU_SETUP_STRU
{
    // mod_cfg define:
    //      bit[7:4]: reserved
    //      bit[3]   : ramp_en -- 0: ramp off, 1: ramp on
    //      bit[2]   : tx_modu -- 0: ook, 1: fsk
    //      bit[1]   : freq_dev_inv -- freq deviation control (0:(+f:1,-f:0), 1:(+f:0,-f:1)
    //      bit[0]   : guass_on -- 0:guass off, 1:guass on
    uint8_t mod_cfg;

    // vco_cfg1 define:
    //      bit[7]   : vco_hband -- pll hband
    //      bit[6]   : pdcplf_cpbias_code -- charge pump bias current control
    //      bit[5:4]: divx_code -- pll divider1(/1/2/3)
    //      bit[3:0]: divx_sel -- pll divider(/1/2/4/8/16)
    uint8_t vco_cfg1;

    // vco_cfg0 define:
    //      bit[7:5] : reserved
    //      bit[4:2]: vco_gain_code -- vco gain code
    //      bit[1:0]: pll_bw_sel -- pll bandwidth control: 0-300khz, 1-210khz
    uint8_t vco_cfg0;

    uint8_t plln_cfg;               // pll n value config
    uint8_t pllk_cfg1;              // pll k value[15:8]
    uint8_t pllk_cfg0;              // pll k value[7:0]

    // pa_cfg define:
    //      bit[7:5]: reserved
    //      bit[4]   : pa_diff_sel -- pa output mode (1:differential mode, 0:single-ended mode)
    //      bit[3]   : pa_rcramp_selb -- pa ramp rc filter select (1: rc filter is not selected, 0: rc filter is selected)
     //      bit[2:0]: pa_ramp_rsel -- select rc filter conrner of pa ramp control
     //                 (0: 640khz, 1: 320kHz, 2: 160kHz, 3: 80kHz, 4: 40kHz, 5: 20kHz, 6: 10kHz, 7: 5kHz)
    uint8_t pa_cfg;

    // pa_idac_code_cfg define:
    //      bit[7:6]: reserved
    //      bit[5:0]: pa_idac_code -- pa current control bits for power control
```

CMOSTEK Confidential

```
    uint8_t pa_idac_code_cfg;
    // ramp_step_time_cfg1 define:
    //      bit[7]    : reserved
    //      bit[6:0]: ramp step time[14:8]
    uint8_t ramp_step_time_cfg1;
    uint8_t ramp_step_time_cfg0;        // ramp step time[7:0]

    uint8_t freq_dev_cfg1;              // freq deviation[15:8]
    uint8_t freq_dev_cfg0;              // freq deviation[7:0]

    uint8_t symbol_time_cfg1;           // symbol time[15:8]
    uint8_t symbol_time_cfg0;           // symbol time[7:0]


} STRU_TX_MODU_SETUP;
```

9.3 Low-frequency Wakeup Module Configuration Structure

Low-frequency wakeup module configuration structure is defined in file CMT216xA.H as follows.

```
typedef struct LFRX_SETUP_STRU
{
    // lfrx_agc_cfg0
    //       bit[7:6] : reserved
    //       bit[5:4] : lfrx_agc_in, AGC Input
    //       bit[3:0] : lfrx_agc_vhref
    uint8_t lfrx_agc_cfg0;

    // lfrx_agc_cfg1
    //       bit[7:6] : reserved
    //       bit[5:4] : lfrx_agc_cnt
    //       bit[3:0] : lfrx_agc_vlref
    uint8_t lfrx_agc_cfg1;

    // lfrx_cadet_cfg
    //       bit[7:6] : lfrx_cadet_win
    //       bit[5:4] : lfrx_cadet_ok_cnt
    //       bit[3:2] : lfrx_peakdet_clk
    //       bit[1:0] : lfrx_data_clk
    uint8_t lfrx_cadet_cfg;
```

CMOSTEK Confidential

```
// lfrx_data_cfg0
//        bit[7:6] : lfrx_data_r0
//        bit[5:4] : lfrx_data_r1
//        bit[3:2] : lfrx_peakdet_c
//        bit[1:0] : lfosc_f_sel
uint8_t lfrx_data_cfg0;

// lfrx_data_cfg1
//        bit[7:4] : lfrx_data_c1
//        bit[3:0] : lfrx_data_c0
uint8_t lfrx_data_cfg1;

// lfrx_data_cfg2
//        bit[7:6] : lfrx_data_c3
//        bit[3:0] : lfrx_data_c2
uint8_t lfrx_data_cfg2;

// lfrx_cmp_cfg0
//        bit[7]   : lfrx_cmp_noise_mask
//        bit[6]   : lfrx_cmp_sw
//        bit[5:3] : lfrx_rssiamp_ibias
//        bit[2:0] : lfrx_pga_ibias
uint8_t lfrx_cmp_cfg0;

// lfrx_cmp_cfg1
//        bit[7:4] : lfrx_cmp_ref
//        bit[3]   : lfrx_demod_th_hold
//        bit[2:0] : lfrx_rssirec_ibias
uint8_t lfrx_cmp_cfg1;

// lfrx_snrdet_cfg0
//        bit[7:6] : lfrx_snrdet_invalid_win
//        bit[5:4] : lfrx_snrdet_valid_win
//        bit[3:0] : lfrx_snrdet_snr
uint8_t lfrx_snrdet_cfg0;

// lfrx_ch_cfg
//        bit[7]   : lfrx_meas_source
//        bit[6:5] : lfrx_osc_vref
//        bit[4]   : lfrx_ch_z
//        bit[3]   : lfrx_ch_y
//        bit[2]   : lfrx_ch_x
//        bit[1:0] : lfrx_startup_manual
```

CMOSTEK Confidential

```
uint8_t lfrx_ch_cfg;

// lfrx_mode_cfg
//        bit[7:6] : lfrx_mode
//        bit[5]   : lfrx_signal_ok_type
//        bit[4:3] : lfrx_timer_extend_mode
//        bit[2]   : duty_cycle_method
//        bit[1]   : lfrx_duty_cycle_en
//        bit[0]   : always_lfrx
uint8_t lfrx_mode_cfg;

// lfrx_timer_cfg0
//        bit[7:3] : lfrx_timer_m_rx_t1
//        bit[2:0] : lfrx_timer_r_rx_t1
uint8_t lfrx_timer_cfg0;

// lfrx_timer_cfg1
//        bit[7:3] : lfrx_timer_m_rx_t2
//        bit[2:0] : lfrx_timer_r_rx_t2
uint8_t lfrx_timer_cfg1;

// lfrx_timer_cfg2
//        bit[7:0] : lfrx_timer_m_sleep
uint8_t lfrx_timer_cfg2;

// lfrx_timer_cfg3
//        bit[7:6] : reserved
//        bit[5]   : lfrx_wakeup_autoclr_dis
//        bit[4:3] : lfrx_wakeup_mode
//        bit[2:0] : lfrx_timer_r_sleep
uint8_t lfrx_timer_cfg3;

// lfrx_snrdet_cfg1
//        bit[7]   : lfrx_rssi_meas_dis
//        bit[6]   : lfrx_dbuf_dis
//        bit[5:3] : lfrx_snrdet_win
//        bit[2:0] : lfrx_meas_win
uint8_t lfrx_snrdet_cfg1;

// lfrx_packet_cfg0
//        bit[7:5] : lfrx_dbuf_length
//        bit[4]   : lfrx_wkid_en
//        bit[3:2] : lfrx_wkid_length
```

CMOSTEK Confidential

```
//      bit[1:0] : lfrx_sync_lenth
uint8_t lfrx_packet_cfg0;

// lfrx_packet_cfg1
//      bit[7:6] : lfrx_ant_mode
//      bit[5]   : lfrx_hold_rst_sel
//      bit[4] : lfrx_snrdet_refin_sel
//      bit[3] : lfrx_man_type
//      bit[2] : lfrx_wkid_man_en
//      bit[1] : lfrx_dig_dataout_sel
//      bit[0] : lfrx_data_man_en
uint8_t lfrx_packet_cfg1;

uint8_t lfrx_sync_value_cfg0;      // lfrx_sync_value_cfg0, bit[7:0]: lfrx_sync_value[7:0]
uint8_t lfrx_sync_value_cfg1;      // lfrx_sync_value_cfg1, bit[7:0]: lfrx_sync_value[15:8]
uint8_t lfrx_sync_value_cfg2;      // lfrx_sync_value_cfg2, bit[7:0]: lfrx_sync_value[23:16]
uint8_t lfrx_sync_value_cfg3;      // lfrx_sync_value_cfg3, bit[7:0]: lfrx_sync_value[31:24]

uint8_t lfrx_wkid_value_cfg0;      // lfrx_wkid_value_cfg0, bit[7:0]: lfrx_wkid_value[7:0]
uint8_t lfrx_wkid_value_cfg1;      // lfrx_wkid_value_cfg1, bit[7:0]: lfrx_wkid_value[15:8]
uint8_t lfrx_wkid_value_cfg2;      // lfrx_wkid_value_cfg2, bit[7:0]: lfrx_wkid_value[23:16]
uint8_t lfrx_wkid_value_cfg3;      // lfrx_wkid_value_cfg3, bit[7:0]: lfrx_wkid_value[31:24]

// lfrx_agc_cfg2
//      bit[7]   : lfrx_dataout_sel
//      bit[6]   : lfrx_decode_seq
//      bit[5]   : lfrx_signal_ok_autoclr_dis
//      bit[4]   : lfrx_agc_en
//      bit[3]   : lfrx_agc_step
//      bit[2:0] : lfrx_agc_cnt_th
uint8_t lfrx_agc_cfg2;

// lfrx_agc_cfg3
//      bit[7:4] : lfrx_dqres
//      bit[3:0] : lfrx_agc_min_index
uint8_t lfrx_agc_cfg3;

// lfrx_agc_cfg4
//      bit[7:4] : lfrx_dr_sel
//      bit[3]   : lfrx_clkgate_dis
//      bit[2]   : lfrx_enable_mode
//      bit[1]   : lfrx_agc_clkgate_dis
//      bit[0]   : lfrx_agc_start_sel
```

CMOSTEK Confidential

```
    uint8_t lfrx_agc_cfg4;

    uint8_t lfrx_cadet_th_h_cfg;        // lfrx_cadet_th_h_cfg, bit[7:0]: lfrx_cadet_th_h
    uint8_t lfrx_cadet_th_l_cfg;        // lfrx_cadet_th_l_cfg, bit[7:0]: lfrx_cadet_th_l

    uint8_t lfrx_signal_ok_clr_th_cfg;  // lfrx_signal_ok_clr_th_cfg, bit[7:0]: lfrx_signal_ok_clr_th

    uint8_t lfrx_data_length_cfg;       // lfrx_data_length_cfg, bit[7:0]: lfrx_data_length

} STRU_LFRX_SETUP;
```

9.4 Low-frequency AFE Module Configuration Structure

9.4.1 Analog Front End Module Configuration Structure

The analog front end module configuration structure (AFE_FRONT_SETUP) is defined in file CMT216xA.H as follows.

```
typedef struct AFE_FRONT_SETUP_STRU
{
    // afe_ia_cfg
    //      bit[7:6] : reserved
    //      bit[5]   : afe_ia_vcmx, AFE IA internal VCM output select 0: vdd/2 (default), 1: vdd/10
    //      bit[4:3] : afe_oa_vcmx, AFE internal VCM input select
    //                 00: vdd/20 (default), 01: vdd/10, 10: vdd/4, 11: vdd/2
    //      bit[2:1] : afe_oa_outx, AFE IA output signal select for SAR ADC input,
    //                 00: OA0 output (default), 01: OA1 output, 10: OA2 output, 11: general purpose external input
    //      bit[0]   : afe_ia1_cx, AFE non-inverted amplification select of the 1st-stage IA,
    //                   0: enable,   1: disable (default)
    uint8_t afe_ia_cfg;

    // afe_oa0_cfg
    //      bit[7]   : afe_oa0_ox, AFE OA0 output signal select,
    //                 0: IA feedback connection (default), 1: output to PAD
    //      bit[6:5] : afe_oa0_nx, AFE OA0 negative input signal select
    //                 00: invalid (default),              01: external input from PAD,
    //                 10: IA feedback connection,         11: buffer connection
    //      bit[4:2] : afe_oa0_px, AFE OA0 positive input signal select,
    //                 000: invalid (default),             001: general purpose measurement channel,
    //                 010: IA feedback connection,        011: OA1 output, 100: OA2 output,
    //                 101: LBD input, 110: diaga input, 111: internal opamp VCM input
    //      bit[1:0] : afe_oa0_na_gx, AFE gain setting for OA0 non-inverted amplification,
    //                 00: invalid (default),      01: x2,     10: x4,     11: x7
```

CMOSTEK Confidential

```
uint8_t afe_oa0_cfg;

// afe_oa1_cfg
//      bit[7]   : sar_direct_dis
 //     bit[6:5] : afe_oa1_ox, AFE OA1 output signal select,
 //               00: invalid (default), 01: output 1 to PAD, 10: output 2 to PAD, 11: IA feedback connection
 //     bit[4:3] : afe_oa1_nx, AFE OA1 negative input signal select,
 //               000: invalid (default),                   001: external negative input 1 from PAD,
 //               010: external negative input 2 from PAD, 011:   IA feedback connection,
 //               100: buffer connection,                   other: reserved
 //     bit[2:0] : afe_oa1_px, AFE OA1 positive input signal select,
 //               000: invalid (default),                  001: external positive input 1 from PAD,
 //               010: external positive input 2 from PAD, 011: OA2 output,
 //               100:   internal opamp VCM input,         other: reserved
uint8_t afe_oa1_cfg;

// afe_oa2_cfg
//      bit[7]   : reserved
 //     bit[6:5] : afe_oa2_ox, AFE OA2 output signal select,
 //               00: invalid (default), 01: output 1 to PAD, 10: output 2 to PAD, 11: IA feedback connection
 //     bit[4:3] : afe_oa2_nx, AFE OA2 negative input signal select,
 //               000: invalid (default),                   001: external negative input 1 from PAD,
 //               010: external negative input 2 from PAD, 011:   IA feedback connection,
 //               100: buffer connection,                   other: reserved
 //     bit[2:0] : afe_oa2_px, AFE OA2 positive input signal select,
 //               000: invalid (default),                   001: external positive input 1 from PAD,
 //               010: external positive input 2 from PAD, 011: OA2 output,
 //               100: internal opamp VCM input,            other: reserved
uint8_t afe_oa2_cfg;

// afe_sar_cfg
//      bit[7:6] : reserved
 //     bit[5:2] : sar_inx, SAR ADC general purpose analog input channel select,
 //                 0000: invalid (default), 0001 - 1001: measurement channel select, other: reserved
 //     bit[1:0] : sar_refx, SAR ADC reference voltage select,
 //               00: LDO output (2.2 V),            01: Bandgap output (1.2V),
 //               10: external reference w/i buffer,  11: external reference w/o buffer
uint8_t afe_sar_cfg;

// afe_sen_chx_cfg
//      bit[7:6] : reserved
 //     bit[5]   : afe_osadj_refx, AFE offset calibration reference select signal,
 //               1: bandgap (about 1.2V),   0: LDO divider output (default)
```

CMOSTEK Confidential

```
    //      bit[4:3] : afe_te_rbx, AFE bridge resistor select for external temperature,
    //               00: invalid (default), 01: 7.0 kohm, 10: 7.5 kohm, 11: 8.0 kohm
    //      bit[2:0] : afe_sen_chx, AFE sensor channel select,
    //               000: invalid (default),              001: external sensor channel 1,
    //               010: external sensor channel 2,      011: internal temperature channel,
    //               other: reserved
    uint8_t afe_sen_chx_cfg;

    // afe_ia_gx_cfg
    //      bit[7:6] : afe_ia2_gx, AFE gain select of the 1st-stage IA,
    //               000: invalid (default),   001: x2,       010: x4,        011: x8,
    //               100: x16,                 101: x24,      110: x32,       111: x48
    //      bit[5:3] : reserved
    //      bit[2:0] : afe_ia1_gx, AFE gain select of the 2nd-stage IA,
    //                 00: x1 (default),       01: x1.5,      10: x2,         11: x4
    uint8_t afe_ia_gx_cfg;

    // afe_ldo_cfg
    //      bit[7:6] : ldo_sar_vo_sel
    //      bit[5]   : ldo_sar_railb
    //      bit[4:0] : reserved
    uint8_t afe_ldo_cfg;

}STRU_AFE_FRONT_SETUP;
```

9.4.2 SNOOZE Function Configuration Structure

The SNOOZE function configuration structure (AFE_FRONT_SETUP) is defined in file CMT216xA.H as follows.

```
typedef struct AFE_SNOOZE_SETUP_STRU
{
    uint8_t snooze_timer_m_value;       // snooze_timer_m_value, snooze time = m*2^(r+1) * 31.25 us

    // snooze_timer_r_value
    //      bit[7:6] : snooze_uth[9:8]
    //      bit[5:4] : snooze_dth[9:8]
    //      bit[3:0] : snooze timer r value
    uint8_t snooze_timer_r_value;
    uint8_t snooze_uth_cfg;             // snooze_uth_cfg, snooze_uth[7:0]
    uint8_t snooze_dth_cfg;             // afe_oa2_cfg, snooze_dth[7:0]

    // snooze_wakeup_cfg
    //      bit[7:4] : reserved
```

CMOSTEK Confidential

```
    //       bit[3]   : dwth_wk_en
    //       bit[2]   : upth_wk_en
    //       bit[1]   : wout_wk_en
    //       bit[0]   : win_wk_en
    uint8_t snooze_wakeup_cfg;

}STRU_AFE_SNOOZE_SETUP;
```

9.4.3 Analog Front End Module Enabling Configuration Structure

Analog front end module enabling configuration structure (AFE_FRONT_POWERUP_SETUP_STRU) is defined in file CMT216xA.H as follows.

```
typedef struct AFE_FRONT_POWERUP_SETUP_STRU
{
    // afe_front_powerup_cfg0
    //       bit[7:5] : reserved
    //       bit[4]   : pd_bg
    //       bit[3]   : sar_lbd_dis
    //       bit[2]   : sar_ref_dis
    //       bit[1]   : pd_ldo_sar
    //       bit[0]   : pd_afe_vtr
    uint8_t afe_front_powerup_cfg0;

    // afe_front_powerup_cfg1
    //       bit[7]   : pd_afe_oacmi
    //       bit[6]   : reserved
    //       bit[5]   : pd_afe_osadj
    //       bit[4]   : pd_afe_iacmo
    //       bit[3]   : pd_sar
    //       bit[2]   : pd_afe_oa2
    //       bit[1]   : pd_afe_oa1
    //       bit[0]   : pd_afe_oa0
    uint8_t afe_front_powerup_cfg1;

}STRU_AFE_FRONT_POWERUP_SETUP;
```

CMOSTEK Confidential

9.5 SPI Interface Configuration Structure

SPI interface configuration structure (SPI_SETUP) is defined in CMT216xA.H as follows.

```
typedef struct SPI_SETUP_STRU
{
      // spi_cfg0 define
      //        bit[7] : bidi_mode      0: 2-line unidir mode,                    1: 1-line bidir mode
      //        bit[6] : bidi_oe        0: bidir mode output disabled (rx-only),  1: bidir mode output enabled (tx-only)
      //        bit[5] : rx_only        0: unidir mode full duplex (rx and tx),   1: unidir mode output disabled (rx-only)
      //        bit[4] : dff            0: 8-bit data frame,                      1: 16-bit data frame
      //        bit[3] : txdmaen  0: tx buffer dma disabled,                1: tx buffer dma enabled
      //        bit[2] : ssoe           0: NSS output disabled in master mode,    1: NSS output enabled in master mode
       //       bit[1] : ssm            0: software control slave NSS disabed,
       //                               1: software control slave NSS enabled (NSS = SSI)
      //        bit[0] : ssi         used to replace NSS in software control slave NSS mode
      uint8_t spi_cfg0;

      // spi_cfg1 define
      //        bit[7]   : lsb_first    0: MSB transmitted first,        1: LSB transmitted first
      //        bit[6]   : reserved
      //        bit[3:5] : br        SPI baud rate, clk_per divided by 0:2, 1:8, 2:16, 3:24, 4:32, 5:64, 6:128, 7:256
      //        bit[2]   : mstr      0: slave mode,                 1: master mode
      //        bit[1]   : cpol      0: SCK is 0 when IDEL,         1: SCK is 1 when IDLE
      //        bit[0]   : cpha    0: first SCK edge to capture first data bit, 1: second SCK edge to capture first data bit
       uint8_t spi_cfg1;

} STRU_SPI_SETUP;
```

9.6 CRC-16 Structure

CRC-16 structure (CRC16_CFG_STRU) is defined in file CMT216xA.H as follows.

```
typedef struct CRC16_CFG_STRU
{
      uint8_t      *pbuf;          // point to the buffer of data which will be crc calc
      uint16_t     data_len;       // the length of data which will be crc calc
      uint16_t     crc_poly;       // the crc16 calc poly
      uint16_t     crc_init;       // the crc16 init value
      uint8_t      bit_order;      // the bit order of data in crc calc, 0: msb first, 1: lsb first

}STRU_CRC16_CFG;
```

CMOSTEK Confidential

9.7 GPIO Output Mapping List

Corresponding to the specific GPIOn pins, the output function configured through GPIOn_OUT_SEL is listed in the below table.

Table 11. 16 Output Function Selection List

Decimal Value	Binary Value	Output Mapping Function	Function Description
0	4b'0000 (default)	gpio_out_r[n]	Control through the register GPIO_OUT_R
1	4b'0001	When n≤7, P0.n When n≥8, P0.(n-8)	Control through the 8051's Port0, supporting bit operation.
2	4b'0010	tb_out0	Timer B capture/compare output 0
3	4b'0011	tb_out1	Timer B capture/compare output 1
4	4b'0100	tb_out2	Timer B capture/compare output 2
5	4b'0101	nss_out	SPI (master mode) chip selection pin
6	4b'0110	sck_out	SPI (master mode) clock signal pin
7	4b'0111	miso_out	SPI (slave mode) MISO signal pin
8	4b'1000	mosi_out	SPI (master mode) MISO signal pin
9	4b'1001	rxd0_out	8051 series port 0 RxD output pin (synchronous serial mode)
10	4b'1010	txd0	8051 serial port 0 TxD output pin
11	4b'1011	ta_out0	Timer A capture/compare output 0
12	4b'1100	ta_out1	Timer A capture/compare output 1
13	4b'1101	ta_out2	Timer A capture/compare output 2
14	4b'1110	led_out_lv	LED function output pin
15	4b'1111	t1_ov	8051 Timer1 overflow signal pin

Notes:

1. In above, n means the current n value of GPIOn. For example, for GPIO0, n is 0; for GPIO8, n is 8.

CMOSTEK Confidential

9.8 Function Module Input Selection GPIO List

For input functions of specific function modules, such as the GPIOn pin, the output function selection is configured through GPIOn_OUT_SEL as listed in the below table.

Table 12. Input Selection GPIO List for Various Function Modules

Decimal Value	Binary Value	GPIO Selection	Corresponding Pin
0	4b’0000 (default)	GPIO0	A0
1	4b’0001	GPIO1	A1
2	4b’0010	GPIO2	A2
3	4b’0011	GPIO3	A3
4	4b’0100	GPIO4	A4
5	4b’0101	GPIO5	A5
6	4b’0110	GPIO6	A6
7	4b’0111	GPIO7	A7
8	4b’1000	GPIO8	B0
9	4b’1001	GPIO9	B1
10	4b’1010	GPIO10	B2
11	4b’1011	GPIO11	B3
12	4b’1100	GPIO12	B4
13	4b’1101	GPIO13	B5
14	4b’1110	GPIO14	B6
15	4b’1111	GPIO15	B7

CMOSTEK Confidential

10 Revise History

Table 13. Revise History Records

Version No.	Chapter	Description	Date
0.4	All	Initial version	2018-05-13
0.4D	All	Reformat and add/delete some APIs	2019-07-01

CMOSTEK Confidential

11 Contacts

CMOSTEK Microelectronics Co., Ltd. Shenzhen Branch
Address: 2/F Building 3, Pingshan Private Enterprise S.T. Park, Xili, Nanshan District, Shenzhen, Guangdong, China

Tel:	+86-755-83231427
Post Code:	518071
Sales:	sales@cmostek.com
Supports:	support@cmostek.com
Website:	www.cmostek.com

CMOSTEK Confidential

Copyright. CMOSTEK Microelectronics Co., Ltd. All rights are reserved.

The information furnished by CMOSTEK is believed to be accurate and reliable. However, no responsibility is assumed for inaccuracies and specifications within this document are subject to change without notice. The material contained herein is the exclusive property of CMOSTEK and shall not be distributed, reproduced, or disclosed in whole or in part without prior written permission of CMOSTEK. CMOSTEK products are not authorized for use as critical components in life support devices or systems without express written approval of CMOSTEK. The CMOSTEK logo is a registered trademark of CMOSTEK Microelectronics Co., Ltd. All other names are the property of their respective owners.