CMOSTEK AN161

CMT2219B Quick Start Guide

Overview

This document provides the CMT2219B register information and usage for users engaging in CMT2219B development. Users can query related register information and usage in this document to guide CMT2219B based 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	Frequency Range	Modulation Method	Chip Function	Configuration Method	Package
CMT2219B	127 - 1020 MHz	(G)FSK/OOK	Receiving	Register	QFN16

CMT2219B is a wireless receiver requiring external MCU control. This document helps users get quick start on CMT2219B development. By using this document along with following application documents, users can get all the information assisting in software and hardware development.

《AN163-CMT2219B RF Parameter Configuration Guide》

《AN164-CMT2219B Low Power Mode User Guide》

《AN165-CMT2219B Operation Guide for Special Features》

《AN166-CMT2219B RSSI User Guide》

《AN167-CMT2219B FIFO and Packet Format Operation Guide》

《AN168-CMT2219B RF-EB User Guide》

《AN197-Quick Manual Frequency Hopping》

《CMT2300A-CMT2219B Frequency Hopping Calculation Table》

《AN198-CMT2300A-CMT2119B-CMT2219B State Switching Considerations》

《AN199-CMT2300A-CMT2119B-CMT2219B RF Frequency Calculation Guide》

Table of Contents

1 Chip Architecture

1.1 How CMT2219B Works

As a digital analog integrated receiver product, the CMT2219B provides reference frequency and digital clock for PLL using a 26 MHz crystal. it supports OOK and (G)FSK demodulation modes. The CMT2219B requires an external MCU to operate the receiver through SPI interface to fulfill various tasks. With MCU operating, it supports both packet and direct data processing methods. The CMT2219B is configured through writing the registers entirely by MCU during receiver power on.

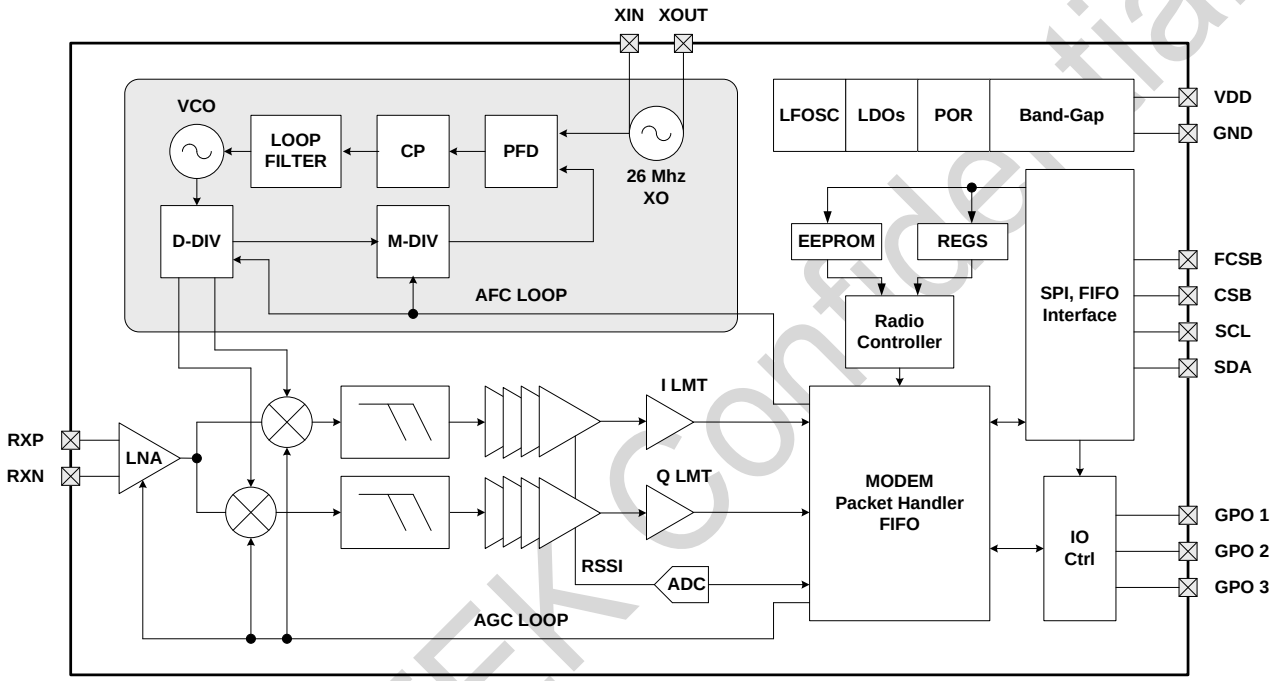


Figure 1. CMT2219B System Block Diagram

The receiver part uses the low intermediate structure of LNA+MIXER+IFFILTER+LIMITTER+PLL to achieve wireless reception at frequencies below 1G.

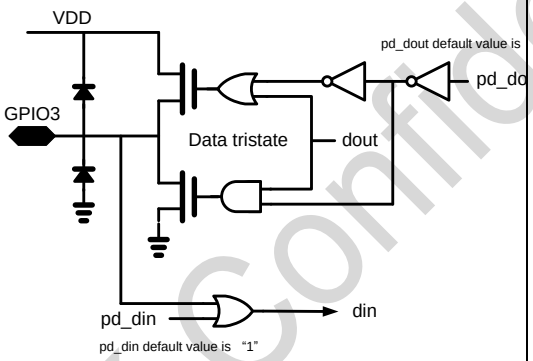
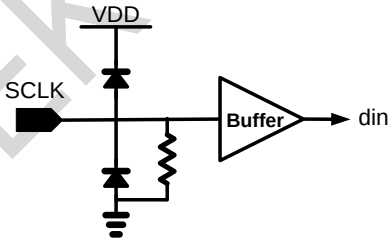
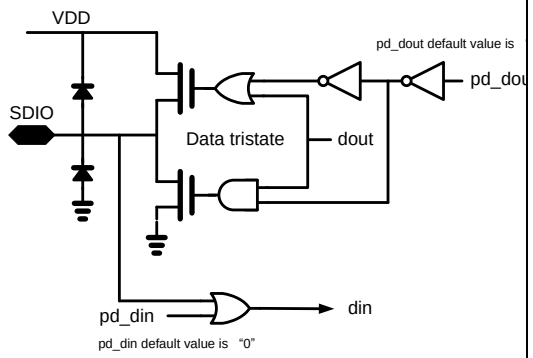
In the receiver system, the analog circuit down-mixes RF signals into intermediate frequencies, and performs digital-to-analog conversion on the intermediate frequency signals through the limiter module, then outputs two single-bit I/Q signals to the digital circuit for the subsequent (G)FSK demodulation. At the same time, the real-time RSSI is converted into an 8-bit digital signal by SARADC and sent to the digital part for subsequent OOK demodulation. The digital circuit down-mixes intermediate frequency signals into zero frequencies (baseband) and performs a series of filtering and decision processing, meanwhile, it controls the analog circuit dynamically through AFC and AGC. Finally a 1-bit original signal is demodulated. The demodulated signal will be sent to the decoder for decoding and then filled in the FIFO, or output to the GPO directly.

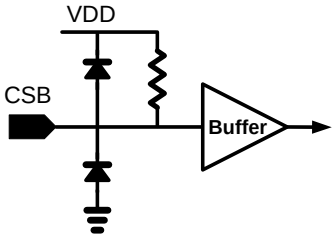
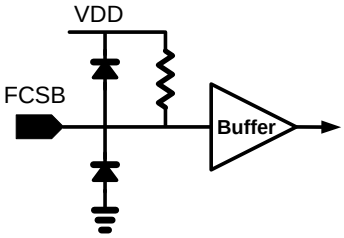
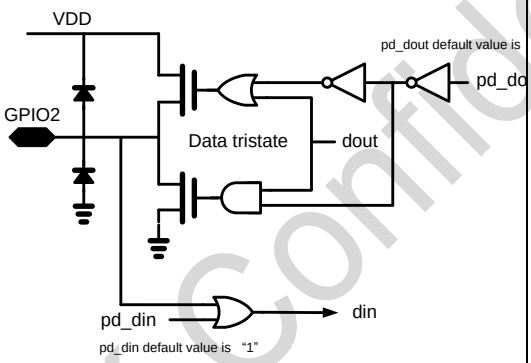
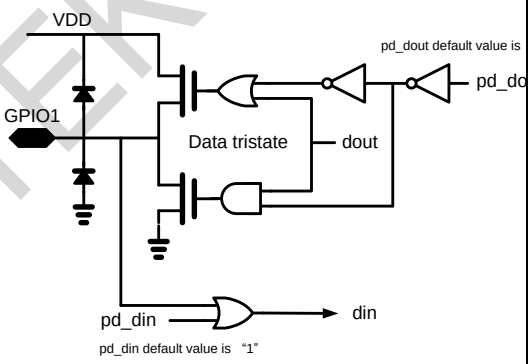
By accessing registers, the external MCU can configure various chip functions, control the master state machine, and access the FIFO through the chip's SPI communication port.

1.2 IO Pin Description

Taking QFN16 as an example, the pin arrangement and function description of the CMT2219B are listed in the below table.

Table 2. CMT2219B Pin Description

Pin #	Pin Name	Type	I/O	Internal IO Circuit Diagram	Description
1	RFIP	Analog	I		RF signal input P
2	RFIN	Analog	I		RF signal input N
3	NA	Analog	O		NA
4	AVDD	Analog	IO		Analog VDD
5	AGND	Analog	IO		Analog GND
6	DGND	Analog	IO		Digital GND
7	DVDD	Analog	IO		Digital VDD
8	GPIO3	Digital	IO		Can be configured as CLKO，DOUT/DIN，INT2 and DCLK (TX/RX).
9	SCLK	Digital	I		The clock of SPI.
10	SDIO	Digital	IO		The data input and output of SPI.

11	CSB	Digital	I		The chip selection for SPI accessing register.
12	FCSB	Digital	I		The chip selection for SPI accessing FIFO.
13	XI	Analog	I		Crystal circuit input.
14	XO	Analog	O		Crystal circuit output.
15	GPIO2	Digital	IO		Can be configured as INT1, INT2, DOUT, DCLK and RF_SWT.
16	GPIO1	Digital	IO		Can be configured as, INT1, INT2, DOUT, DCLK and RF_SWT.

Notes:

1. INT1 and INT2 are interrupts.
2. DOUT is the demodulation output.
3. DIN is the modulation input.
4. DCLK is the synchronous clock of modulation or demodulation data rate, which is switched automatically when Tx/Rx mode switching occurs.
5. RF_SWT is the control signal for the antenna switch.

2 SPI Interface Timing

The CMT2219B communicates with outside through a 4-wire SPI port. The low active CSB is the chip selection signal used to access the registers. The low active FCSB is the chip selection signal used to access FIFO. They cannot be set both low at the same time. SCLK is a serial clock with speed up to 5 MHz. For both the chip itself or external MCU, data is sent on the falling edge of SCLK and collected on the rising edge. SDIO is a bidirectional pin for inputting and outputting data. Both address and data parts are transmitted starting from the MSB.

2.1 Reading/Writing Register Operation

When accessing registers, the CSB is pulled low. It sends an R/W bit first followed by a 7-bit register address. After the external MCU pulls the CSB low, it must wait for at least half a SCLK cycle before starting transmitting R/W bits. After the MCU sends the falling edge of the last SCLK, it must wait for at least half a CLK cycle before pulling CSB high.

It should be noted that, for the read register operation in Figure 2, both the MCU and CMT2219B will generate a switching IO (SDIO) port action between address 0 and data 7. At this time, the CMT2219B will switch the IO port from input to output, and the MCU will switch the IO port from output to input accordingly. Please be noted the dotted line in the middle, where it is strongly recommended that the MCU switches the IO port to input before sending the falling edge of SCL. The CMT2219B will switch the IO to output after getting the falling edge. By doing so, it can avoid SDIO is set as output by both the MCU and CMT2219B, which will cause electrical conflicts. For some MCUs, such a situation may cause MCU reset or other abnormal behaviors.

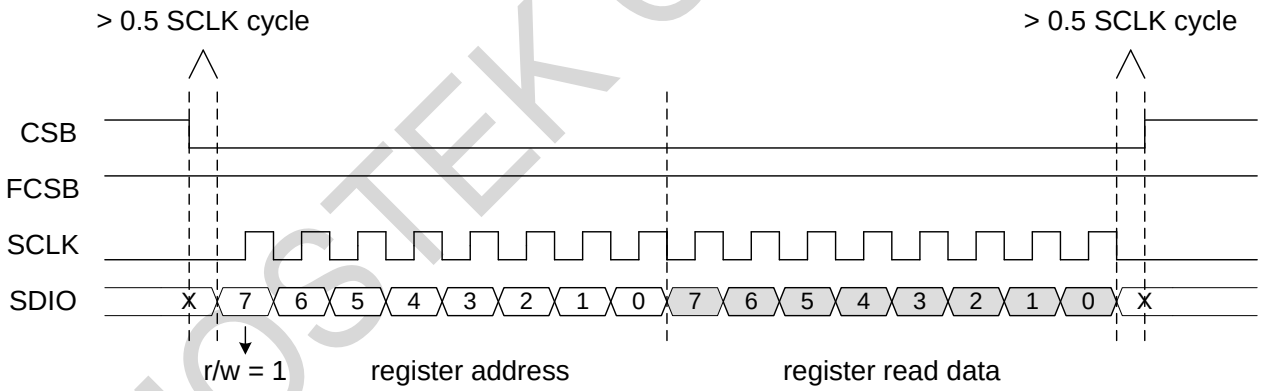


Figure 2. SPI Reading Register Timing

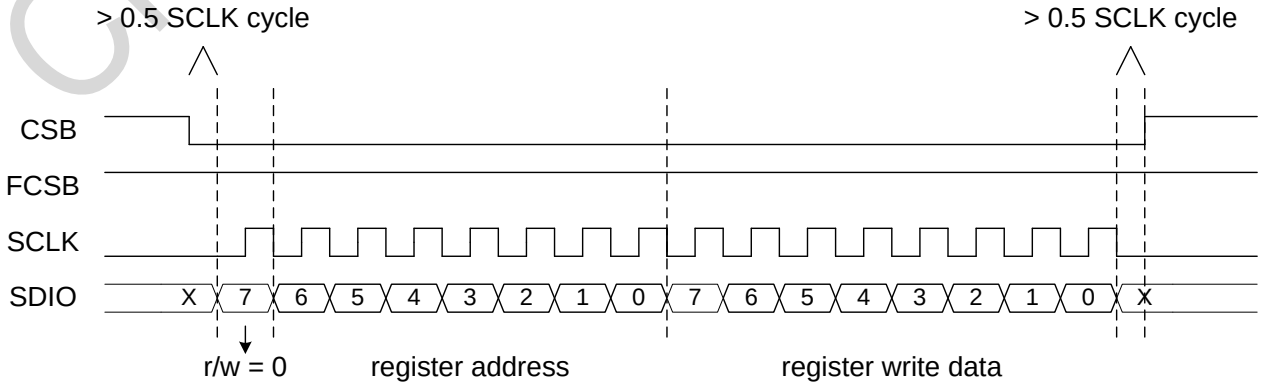


Figure 3. SPI Writing Register Timing

Refer to Appendix 1 for a SPI read operation code example.

2.2 Reading FIFO Operation

When the MCU needs to access the FIFO, it requires to configure some registers first to set the FIFO access mode as well as some other operating modes, which will be discussed in *AN167-CMT2219B FIFO* and *Packet Format Usage Guide*. The timing diagram is shown in the below figure. It should be noted that the control of FCSB and that of CSB are slightly different when accessing the registers. At the beginning of the access, it must pull down the FCSB for one clock cycle before sending the rising edge of SCLK. After sending the falling edge of the last SCL, it needs to wait for at least 2 us then pull up the FCSB. The FCSB must be pulled high for at least 4 us between two consecutive read operations.

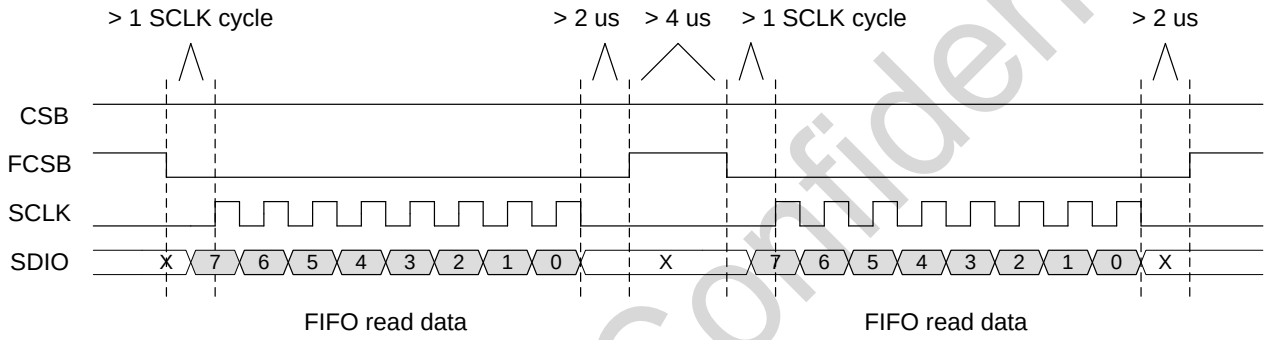


Figure 4. SPI Reading FIFO Timing

3 Configuration and Control Mechanism

3.1 Register Overview

Table 3. CMT2219B Register Overview

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Function
0x00	RW	CUS_CMT1	Import RFPDK generated configuration directly. Users do not need to understand them.								CMT area
0x01	RW	CUS_CMT2									
0x02	RW	CUS_CMT3									
0x03	RW	CUS_CMT4									
0x04	RW	CUS_CMT5									
0x05	RW	CUS_CMT6									
0x06	RW	CUS_CMT7									
0x07	RW	CUS_CMT8									
0x08	RW	CUS_CMT9									
0x09	RW	CUS_CMT10									
0x0A	RW	CUS_CMT11									
0x0B	RW	CUS_RSSI									
Addr	**R/W**	**Name**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Function**
0x0C	RW	CUS_SYS1	LMT_VTR [1:0]		MIXER_BIAS [1:0]		LNA_MODE [1:0]		LNA_BIAS [1:0]		System area
0x0D	RW	CUS_SYS2	LFOSC_RECAL_EN	LFOSC_CAL1_EN	LFOSC_CAL2_EN	RX_TIMER_EN	SLEEP_TIMER_EN	RESV	RX_DC_EN	DC_PAUSE	
0x0E	RW	CUS_SYS3	SLEEP_BYPASS_EN	XTAL_STB_TIME [2:0]			RESV [1:0]		RX_EXIT_STATE [1:0]		
0x0F	RW	CUS_SYS4	SLEEP_TIMER_M [7:0]								
0x10	RW	CUS_SYS5	SLEEP_TIMER_M [10:8]				SLEEP_TIMER_R [3:0]				
0x11	RW	CUS_SYS6	RX_TIMER_T1_M [7:0]								
0x12	RW	CUS_SYS7	RX_TIMER_T1_M [10:8]				RX_TIMER_T1_R [3:0]				
0x13	RW	CUS_SYS8	RX_TIMER_T2_M [7:0]								
0x14	RW	CUS_SYS9	RX_TIMER_T2_M [10:8]				RX_TIMER_T2_R [3:0]				
0x15	RW	CUS_SYS10	COL_DET_EN	COL_OFS_SEL	RX_AUTO_EXIT_DIS	DOUT_MUTE	RX_EXTEND_MODE [3:0]				
0x16	RW	CUS_SYS11	PJD_TH_SEL	CCA_INT_SEL [1:0]		RSSI_DET_SEL [1:0]		RSSI_AVG_MODE [2:0]			
0x17	RW	CUS_SYS12	PJD_WIN_SEL [1:0]		RESV [5:0]						
Addr	**R/W**	**Name**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Function**
0x18	RW	CUS_RF1	Import RFPDK generated configuration directly. Users do not need to understand them.								Frequency area
0x19	RW	CUS_RF2									
0x1A	RW	CUS_RF3									
0x1B	RW	CUS_RF4									
0x1C	RW	CUS_RF5									
0x1D	RW	CUS_RF6									
0x1E	RW	CUS_RF7									
0x1F	RW	CUS_RF8									
Addr	**R/W**	**Name**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Function**
0x20	RW	CUS_RF9	Import RFPDK generated configuration directly. Users do not need to understand them.								Data rate area
0x21	RW	CUS_RF10									
0x22	RW	CUS_RF11									
0x23	RW	CUS_RF12									
0x24	RW	CUS_FSK1									
0x25	RW	CUS_FSK2									
0x26	RW	CUS_FSK3									
0x27	RW	CUS_FSK4									
0x28	RW	CUS_FSK5									
0x29	RW	CUS_FSK6									
0x2A	RW	CUS_FSK7									
0x2B	RW	CUS_CDR1									
0x2C	RW	CUS_CDR2									
0x2D	RW	CUS_CDR3									
0x2E	RW	CUS_CDR4									
0x2F	RW	CUS_AGC1									
0x30	RW	CUS_AGC2									
0x31	RW	CUS_AGC3									
0x32	RW	CUS_AGC4									
0x33	RW	CUS_OOK1									
0x34	RW	CUS_OOK2									
0x35	RW	CUS_OOK3									
0x36	RW	CUS_OOK4									
0x37	RW	CUS_OOK5									
Addr	**R/W**	**Name**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Function**
0x38	RW	CUS_PKT1	RX_PREAM_SIZE [4:0]					PREAM_LENG_UNIT	DATA_MODE [1:0]		Baseband area
0x39	RW	CUS_PKT2	RESV [7:0]								
0x3A	RW	CUS_PKT3	RESV [7:0]								
0x3B	RW	CUS_PKT4	PREAM_VALUE [7:0]								
0x3C	RW	CUS_PKT5	RESV	SYNC_TOL [2:0]			SYNC_SIZE [2:0]			SYNC_MAN_EN	
0x3D	RW	CUS_PKT6	SYNC_VALUE [7:0]								
0x3E	RW	CUS_PKT7	SYNC_VALUE [15:8]								
0x3F	RW	CUS_PKT8	SYNC_VALUE [23:16]								
0x40	RW	CUS_PKT9	SYNC_VALUE [31:24]								
0x41	RW	CUS_PKT10	SYNC_VALUE [39:32]								
0x42	RW	CUS_PKT11	SYNC_VALUE [47:40]								
0x43	RW	CUS_PKT12	SYNC_VALUE [55:48]								
0x44	RW	CUS_PKT13	SYNC_VALUE [63:56]								
0x45	RW	CUS_PKT14	RESV	PAYLOAD_LENG [10:8]			AUTO_ACK_EN	NODE_LENG_POS_SEL	PAYLOAD_BIT_ORDER	PKT_TYPE	
0x46	RW	CUS_PKT15	PAYLOAD_LENG [7:0]								
0x47	RW	CUS_PKT16	RESV	RESV	NODE_FREE_EN	NODE_ERR_MASK	NODE_SIZE [1:0]		NODE_DET_MODE [1:0]		
0x48	RW	CUS_PKT17	NODE_VALUE [7:0]								
0x49	RW	CUS_PKT18	NODE_VALUE [15:8]								
0x4A	RW	CUS_PKT19	NODE_VALUE [23:16]								
0x4B	RW	CUS_PKT20	NODE_VALUE [31:24]								
0x4C	RW	CUS_PKT21	FEC_TYPE	FEC_EN	CRC_BYTE_SWAP	CRC_BIT_INV	CRC_RANGE	CRC_TYPE [1:0]		CRC_EN	
0x4D	RW	CUS_PKT22	CRC_SEED [7:0]								
0x4E	RW	CUS_PKT23	CRC_SEED [15:8]								
0x4F	RW	CUS_PKT24	CRC_BIT_ORDER	WHITEN_SEED [8]	WHITEN_SEED_TYPE	WHITEN_TYPE [1:0]		WHITEN_EN	MANCH_TYPE	MANCH_EN	
0x50	RW	CUS_PKT25	WHITEN_SEED [7:0]								
0x51	RW	CUS_PKT26	RESV [7:0]								
0x52	RW	CUS_PKT27	RESV [7:0]								
0x53	RW	CUS_PKT28	RESV [7:0]								
0x54	RW	CUS_PKT29	RESV	FIFO_TH [6:0]							
Addr	**R/W**	**Name**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Function**
0x55	RW	CUS_RESV1	Reserved area, users do not need to write in								Reserved area
0x56	RW	CUS_RESV1									
0x57	RW	CUS_RESV1									
0x58	RW	CUS_RESV1									
0x59	RW	CUS_RESV1									
0x5A	RW	CUS_RESV1									
0x5B	RW	CUS_RESV1									
0x5C	RW	CUS_RESV1									
0x5D	RW	CUS_RESV1									
0x5E	RW	CUS_RESV1									
Addr	**R/W**	**Name**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Function**
0x5F	RW	CUS_LBD	LBD_TH [7:0]								LBD area

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Function
0x60	RW	CUS_MODE_CTL	CHIP_MODE_SWT [7:0]								
0x61	RW	CUS_MODE_STA	RESV [1:0]		RSTN_IN_EN	CFG_RETAIN	CHIP_MODE_STA [3:0]				
0x62	RW	CUS_EN_CTL	RESV [1:0]		RESV	FIFO_AUTO_CLR_DIS	RESV [3:0]				
0x63	RW	CUS_FREQ_CHNL	FH_CHANNEL [7:0]								
0x64	RW	CUS_FREQ_OFS	FH_OFFSET [7:0]								
0x65	RW	CUS_IO_SEL	RESV [1:0]		GPIO3_SEL [1:0]		GPIO2_SEL [1:0]		GPIO1_SEL [1:0]		Control area 1
0x66	RW	CUS_INT1_CTL	RESV [1:0]		INT_POLAR	INT1_SEL [4:0]					
0x67	RW	CUS_INT2_CTL	RESV	LFOSC_OUT_EN	RESV	INT2_SEL [4:0]					
0x68	RW	CUS_INT_EN	SL_TMO_EN	RX_TMO_EN	RESV	PREAM_OK_EN	SYNC_OK_EN	NODE_OK_EN	CRC_OK_EN	PKT_DONE_EN	
0x69	RW	CUS_FIFO_CTL	RESV [2:0]			FIFO_AUTO_CLR_DIS	RESV [1:0]		FIFO_MERGE_EN	RESV	
0x6A	W	CUS_INT_CLR1	RESV [1:0]		SL_TMO_FLG	RX_TMO_FLG	RESV [1:0]		SL_TMO_CLR	RX_TMO_CLR	
Addr	**R/W**	**Name**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Function**
0x6B	W	CUS_INT_CLR2	RESV [1:0]		LBD_CLR	PREAM_OK_CLR	SYNC_OK_CLR	NODE_OK_CLR	CRC_OK_CLR	PKT_DONE_CLR	
0x6C	W	CUS_FIFO_CLR	RESV [4:0]					FIFO_RESTORE	FIFO_CLR_RX	RESV	
0x6D	R	CUS_INT_FLAG	LBD_FLG	COL_ERR_FLG	PKT_ERR_FLG	PREAM_OK_FLG	SYNC_OK_FLG	NODE_OK_FLG	CRC_OK_FLG	PKT_OK_FLG	Control area 2
0x6E	R	CUS_FIFO_FLAG	RESV	RX_FIFO_FULL_FLG	RX_FIFO_NMTY_FLG	RX_FIFO_TH_FLG	RX_FIFO_OVF_FLG	RESV [2:0]			
0x6F	R	CUS_RSSI_CODE	RSSI_CODE [7:0]								
0x70	R	CUS_RSSI_DBM	RSSI_DBM [7:0]								
0x71	R	CUS_LBD_RESULT	LBD_RESULT [7:0]								

The configuration and control from the external MCU are performed through the access of a series of registers via SPI interface. As seen from the above table, the address ranges from 0x00 to 0x71, which can be logically divided into 3 large areas for easy-understanding, that is, configuration area (consisting of 7 sub-areas, including reserved area), control area 1 and control area 2. The details are as follows.

First of all, for the 3 areas, the address is continuous and the operation mode has no essential difference, meaning that it performs read and write operations following the SPI register accessing timing. However, in terms of function and usage, the 3 areas covers different functions as listed in the below table.

Table 4. CMT2219B Register Area Partitioning

Area	Address	Description
Configuration area	0x00– 0x5F	This area covers the following configurations. 1. hidden attributes used internally by CMT 2. system operation 3. RF characteristics 4. FSK demodulation 5. clock recovery 6. AGC characteristics 7. OOK demodulation 8. packet format and encoding and decoding mode The register values come from either RFPDK or user applications which change the register values based on specific application requirements. Generally, most of the registers only need to be configured once during the initialization process except some individual configurations on RF frequency or data rate, which may need to be configured multiple times in applications.
Control area 1	0x60 – 0x6A	This area is used to control the chip per real time base in applications, coving the following control functions. 1. chip state switching 2. configuration of special functions 3. manual frequency hopping for channel switching 4. IO control 5. Interrupt enabling 6. FIFO operating mode configuration 7. control of chip status related interrupts Control area 1 can be saved in the SLEEP state. The configured register values will not be

Area	Address	Description
		lost if no chip reset or battery power loss occurs.
Control area 2	0x6B – 0x71	This area is provided for users to control the chip, covering the following control functions. 1. the control of closing interrupt first for packet and FIFO 2. FIFO control 3. reading of RSSI values 4. control of LBD function Control area 2 cannot be saves in the SLEEP state. Pls. be noted that all configured values and readable values will disappear in SLEEP state.

3.2 Operating State Switching

The external MCU can switch and query the chip operating state by accessing registers in control areas 1 as listed in the below table.

Table 5. Registers for State Control

Name	Bits	R/W	Flag	Description
CUS_MODE_CTL (0x60)	7:0	RW	CHIP_MODE_SWT<7:0>	State control commands. 00000010: go_stby 00000100: go_rfs 00001000: go_rx 00010000: go_sleep Other values: not allow to send command.
CUS_MODE_STA (0x61)	3:0	RW	CHIP_MODE_STA<3:0>	Chip state. 0000: IDLE 0001: SLEEP 0010: STBY 0011: RFS 0101: Rx 1000: ERROR 1001: CAL Other values: invalid The LOCKING state represents that the PLL is locked. When the lock state ends, it will enter Tx/Rx state. The CAL state represents the calibration state. The chip will not stay in calibration state for a long time, so users usually cannot query out the CAL state.

As seen from the below state switching diagram, the LOCKING and CAL states do not appear in the figure, since they both appear in short procedures under normal circumstances, users may not be able to query out them through the serial port (it's still possible that users can query them out depending on the serial port speed). Users must force the LOCKING_EN register in the control area to 1 during initialization to make the LOCKING status valid. If LOCKING_EN is set to 0, meaning that there is no LOCKING state, the system by default waits for 100 us then enters the RX state, however the PLL may not be locked yet in the case, maybe resulting in a transmission or reception error. Therefore, LOCKING_EN must be set to 1 in normal conditions.

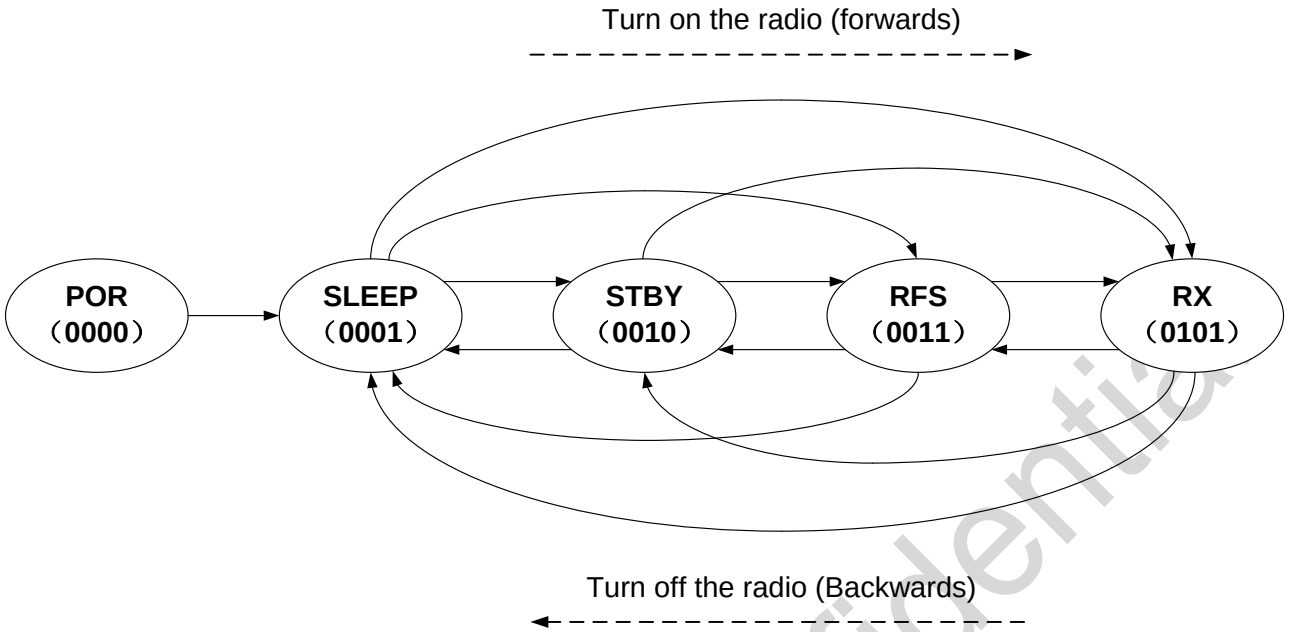


Figure 5. CMT2219B State Transition Diagram

Table 6. CMT2219B State and Operating Modules

State	Binary Code	Switching Command	Operating Module	Optional Module
IDLE	0000	soft_rst	SPI, POR	None
SLEEP	0001	go_sleep	SPI, POR, FIFO	LFOSC, sleep timer
STBY	0010	go_stby	SPI, POR, XTAL, FIFO	None
RFS	0011	go_rfs	SPI, POR, XTAL, PLL, FIFO	None
RX	0101	go_rx	SPI, POR, XTAL, PLL, LNA+MIXER+IF, FIFO	Rx Timer

After the MCU sends a *go_** command, the chip sometimes needs to wait for a certain amount of time to switch states successfully. The followings will discuss each state and the corresponding waiting time before state switching.

● IDLE state and power-on process

After chip power up on VDD, it usually takes about 1 ms to wait for POR release. After POR release, it takes a time between 200 us and 1 ms (depending on the crystal characteristics) to wait for crystal startup. After crystal startup, it needs to wait for the stabilization of the frequency and period of output clock (the stabilization time depends on the crystal characteristics). Users can set the time by writing XTAL_STB_TIME <2:0> (this set time should be longer than the crystal stabilization time) with the default value as 2.48 ms. In general it's not easy for users to observe the required crystal stabilization time, therefore set it to the longest value - 2.48 ms to cover most of crystal types .

A chip stays in the IDLE state before the crystal being stable. When crystal being stable, the chip leaves IDLE and begins calibration for each module. After module calibration, it stays in SLEEP waiting for initialization configuration from users. At any time, once a soft reset is performed, the chip will return to IDLE and re-run power-up process.

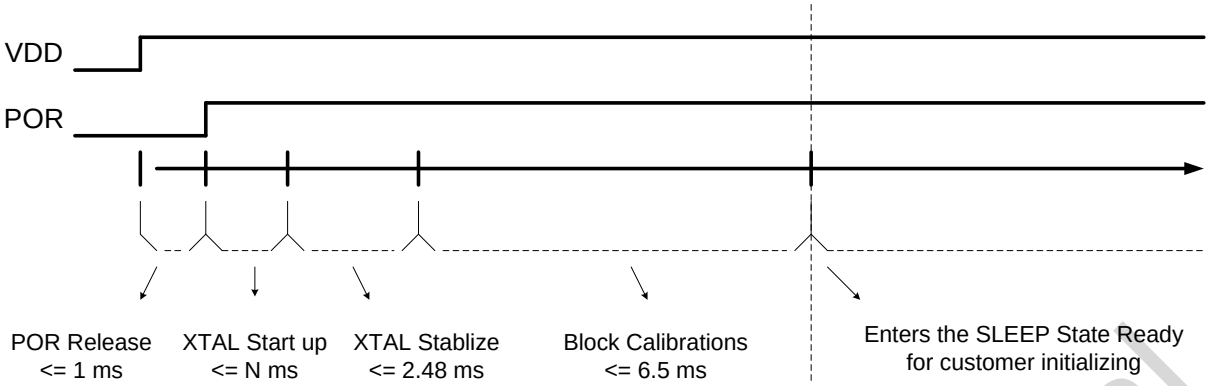


Figure 6. CMT2219B Power ON Process Flow

The module calibrations includes LFOSC calibration to ensure the frequency error of LFOSC is less than 1%. As LFOSC is mainly used to drive the sleep timer, if sleep timer is not used or it doesn't matter using a sleep timer with lower accuracy, users can turn off the LFOSC calibration by setting the LFSOC_CAL1_EN and LFOSC_CAL2_EN to 0 during the configuration phase (see the followings for details). Then after the next soft reset, the LFOSC calibration function will be turned off, saving about 5 ms of calibration time. Anyway, a LFOSC calibration will be performed by default in power up process.

- **SLEEP state**

The chip power consumption in SLEEP is the lowest with almost all modules turned off. In SLEEP state, SPI is enabled, the registers in the configuration area and control area 1 can be saved, and the filled in contents in FIFO remain unchanged. However, users can neither operate FIFO nor change the contents of the registers. In SLEEP state, if a periodic wake-up function is enabled, the LFOSC and sleep counter will start working. Switching from IDLE to SLEEP has been described in the above power-up process. Switching from the rest of the states to SLEEP is done immediately.

- **STBY state**

In STBY, crystal and LDO are turned on, current is slightly increased and FIFO can be operated. Users can choose whether to output CLKO (system clock) to a PIN. As crystal and LDO are turned on, the switching from STBY to Rx will be quicker compared to that from SLEEP. Switching from SLEEP to STBY is performed after crystal startup and stabilization. Switching from the rest of the states to STBY is done immediately.

- **RFS status**

RFS is a transitional state before switching to RX. In this state, all modules are turned on except the receiver's RF module. The current is larger than that in STBY. Switching from STBY to RFS requires approximately 350 us including PLL calibration time and settling time. The time for switching from SLEEP to RFS includes crystal startup and stabilization time. Switching from the rest of the states to RFS is done immediately.

- **RX status**

In Rx, all receiver related modules are turned on. Switching from RFS to RX takes only 20 us. Switching from STBY to RX needs an additional 350 us time for PLL calibration and stabilization. Switching from SLEEP to RX needs an additional time for crystal startup and stabilization.

- **LOCKING State**

This state is specific for PLL locking with a locking time ranging from 50 to 200 us. Users need to configure the LOCKING_EN bit

to 1 during initialization phase. Moreover, due to complicated chip application environments, such conditions as power supply mutation, sporadic strong static electricity and electromagnetic interference may result in chip abnormal cases. In these cases, after users send go_rx or go_rfs commands, the CMT2219B may fail in locking and thus stay in the LOCKING state for a long time. At this time, users can send go_stby command (at this moment, the receiver can recognize go_stby command only) to return to the STBY state. After then users can send commands for state switching or send a soft reset command to reset the CMT2219B.

See Appendix 3 for state a switching function example code.

Please refer to *AN198-CMT2300A-CMT2119B-CMT2219B Status Switching Considerations* for state switching considerations.

3.3 Soft Reset (Softrst)

Users perform a soft reset of the CMT2219B by writing 0xFF to address 0x7F via SPI. Once receiving this command, the chip it will perform a reset operation immediately, returning to the IDLE state then starting chip initialization process immediately. Therefore, after a soft reset command is sent, the IDLE state keeps a very short time. Users can hardly query out this state.

3.4 Introduction to RFPDK

RFPDK is a windows based software tool provided by CMOSTEK for RF chip configuration or programming. For CMT2219B, users input configuration parameters in RFPDK screen, after then a register configuration file is generated by RFPDK. Users can import the register file into MCU program for register configuration. The overall configuration flow is that, during chip initialization phase, it fulfills chip initialization configuration using the register configuration file generated by RFPDK; then in chip operating phase, users can configure and control specific registers flexibly. Therefore, by using RFPDK, users only need to understand RFPDK operations and user application related registers. A RFPDK screen is shown in the below figure. How RFPDK co-works with the registers will be discussed below.

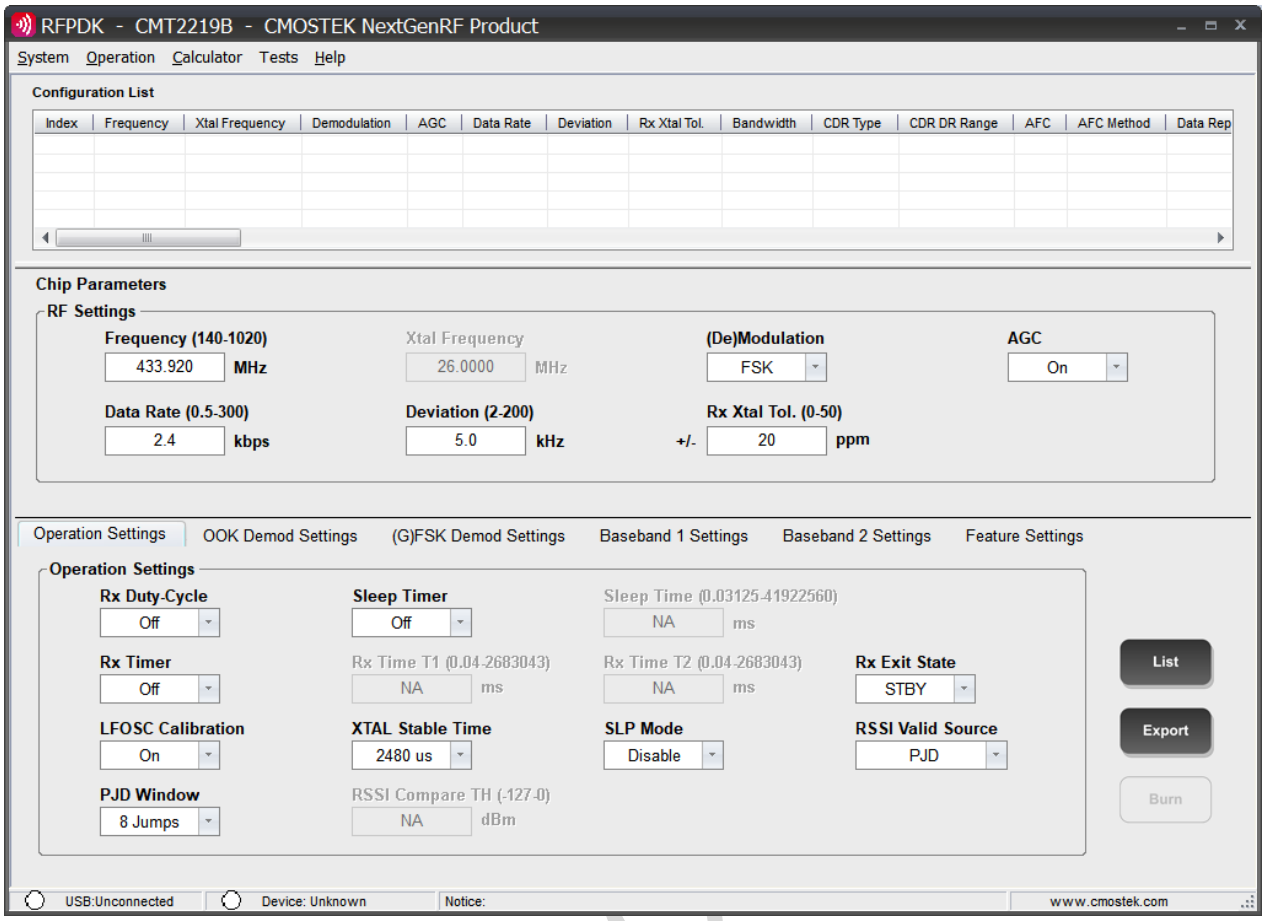


Figure 7. CMT2219B RFPDK

In RFPDK screen, users need to configure 7 major areas as listed in the below table.

Table 7. Major Configuration Areas in RFPDK

RFPDK Area	Configuration Parameter	Relationship Between RFPDK Configuration Parameter and Registers
RF Settings	RF parameters such as frequency and data rate	Register values are generated based on complex calculations on input parameters. Users do not need to understand register details.
Operation Settings	System operating parameters00	The input parameters correspond to the registers one by one. Users can understand register details.
OOK Demod Settings	OOK demodulation parameters	Register values are generated based on complex calculations on input parameters. Users do not need to understand register details. Default values are used in general.
FSK Demod Settings	FSK demodulation parameters	Register values are generated based on complex calculations on input parameters. Users do not need to understand register details. Default values are used in general.
Baseband 1 Settings	Parameters such as baseband codec and FIFO	The input parameters correspond to the registers one by one. Users can understand register details.
Baseband 2 Settings	Baseband packet format parameters	The input parameters correspond to the registers one by one. Users can understand register details.

Feature Settings	parameters for other functional	The input parameters correspond to the registers one by one. Users can understand register details.

Below is an example of the generated configuration file.

The comment at the top of the file is the configuration file generation information, which is used for error checking. Users can ignore this part generally.

```
;------------------------------------------------------------------------
;   CMT2219B Configuration File
;   Generated by CMOSTEK RFPDK 1.46 Beta
;   2017.10.12 15:29
;------------------------------------------------------------------------
; Mode                      = Advanced
; Part Number               = CMT2219B
; Frequency                 = 433.920 MHz
; Xtal Frequency            = 26.0000 MHz
; Demodulation              = FSK
; AGC                       = On
; Data Rate                 = 2.4 kbps
; Deviation                 = 5.0 kHz
; Rx XtalTol.               = 20 ppm
; Bandwidth                 = Auto-Select kHz
; CDR Type                  = Counting
; CDR DR Range              = NA
; AFC                       = On
; AFC Method                = Auto-Select
; Data Representation       = 0:F-low 1:F-high
; Rx Duty-Cycle             = Off
; Sleep Timer               = Off
; Sleep Time                = NA
; Rx Timer                  = Off
; Rx Time T1                = NA
; Rx Time T2                = NA
; Rx Exit State             = STBY
; SLP Mode                  = Disable
; RSSI Valid Source         = PJD
; PJD Window                = 8 Jumps
; LFOSC Calibration         = On
; Xtal Stable Time          = 2480 us
; RSSI Compare TH           = NA
; Data Mode                 = Packet
; Whitening                 = Disable
; Whiten Type               = NA
; Whiten Seed Type          = NA
; Whiten Seed               = NA
; Manchester                = Disable
; Manchester Type           = NA
; FEC                       = Disable
; FEC Type                  = NA
; Packet Type               = Fixed Length
; Node-Length Position      = NA
; Payload Bit Order         = Start from msb
; Preamble Rx Size          = 2
; Preamble Value            = 170
; Preamble Unit             = 8-bit
; Sync Size                 = 2-byte
; Sync Value                = 11732
; Sync Tolerance            = None
; Sync Manchester           = Disable
; Node ID Size              = NA
```

; Node ID Value = NA
; Node ID Mode = None
; Node ID Err Mask = Disable
; Node ID Free = Disable
; Payload Length = 32
; CRC Options = None
; CRC Seed = NA
; CRC Range = NA
; CRC Swap = NA
; CRC Bit Invert = NA
; CRC Bit Order = NA
; Dout Mute = Off
; Dout Adjust Mode = Disable
; Dout Adjust Percentage = NA
; Collision Detect = Off
; Collision Detect Offset = NA
; RSSI Detect Mode = Always
; RSSI Filter Setting = No Filtering
; RF Performance = High
; LBD Threshold = 2.4 V
; RSSI Offset = 26
; RSSI Offset Sign = 1

The second half of the file is the register configuration content that users needs to import into the MCU program, expressed in hexadecimal. For users' convenience, the tool will automatically divide the contents of all registers into 7 sections with address ranging from 0x00 to 0x5F. For easy reading, the content of 6 areas (the area of 0x55 – 0x5F which users can ignore is excluded) are shown in the below list.

;---
; The following are the Register contents
;---

[CMT Bank]

Addr	Value
0x00	0x02
0x01	0x66
0x02	0xEC
0x03	0x1C
0x04	0xF0
0x05	0x80
0x06	0x14
0x07	0x08
0x08	0x91
0x09	0x02
0x0A	0x02
0x0B	0xD0

[System Bank]

Addr	Value
0x0C	0xAE
0x0D	0xE0
0x0E	0x70
0x0F	0x00
0x10	0x00
0x11	0xF4
0x12	0x10
0x13	0xE2
0x14	0x42
0x15	0x20
0x16	0x00
0x17	0x81

[Frequency Bank]

Addr	Value
0x18	0x42
0x19	0x71
0x1A	0xCE
0x1B	0x1C
0x1C	0x42
0x1D	0x5B
0x1E	0x1C
0x1F	0x1C

[Data Rate Bank]

Addr	Value
0x20	0x32
0x21	0x18
0x22	0x00
0x23	0x99
0x24	0xC1
0x25	0x9B
0x26	0x06
0x27	0x0A
0x28	0x9F
0x29	0x39
0x2A	0x29
0x2B	0x29
0x2C	0xC0
0x2D	0x51
0x2E	0x2A
0x2F	0x53
0x30	0x00
0x31	0x00
0x32	0xB4
0x33	0x00
0x34	0x00
0x35	0x01
0x36	0x00
0x37	0x00

[Baseband Bank]

Addr	Value
0x38	0x12
0x39	0x08
0x3A	0x00
0x3B	0xAA
0x3C	0x02
0x3D	0x00
0x3E	0x00
0x3F	0x00
0x40	0x00
0x41	0x00
0x42	0x00
0x43	0xD4
0x44	0x2D
0x45	0x00
0x46	0x1F
0x47	0x00
0x48	0x00
0x49	0x00
0x4A	0x00
0x4B	0x00
0x4C	0x00
0x4D	0x00
0x4E	0x00
0x4F	0x60
0x50	0xFF
0x51	0x01

[LBD Bank]

Addr	Value
0x5F	0x7F

```
0x52   0x00
0x53   0x1F
0x54   0x10
```

The last part is the CRC value used for the tool's internal check, which users can ignore.

```
;--------------------------------------------------------------------------------
;   The following is the CRC result for
;   the above Register contents
;--------------------------------------------------------------------------------
```

0x0E6A

3.5 Chip Initialization Process

Based on users' input parameters in RFPDK, the RFPDK generates the configuration content for configuration area (0x00 – 0x5F). Upon chip powered on or reset, it automatically enters SLEEP waiting for MCU operation. The MCU can operate according to the following initialization process.

1. When the MCU is ready to operate, it sends a soft reset and waits for 20 ms.
2. Confirm that reset completes and stay in SLEEP state.
3. Send go_stby command and confirm the chip enters STBY state.
4. Write the register contents generated by RFPDK into the configuration area with address range as 0x00 - 0x5F.
5. User needs to configure 0x09 CUS_CMT10<2:0> (the 3 bits in the configuration area) to 0b010 with the other bits of this register remain unchanged. If RFPDK version 1.46 or higher version is used, this step can be ignored.
6. If SLEEP TIMER is not required in applications, configure RECAL_LFOSC_EN, LFSOC_CAL1_EN and LFOSC_CAL2_EN to 0.
7. If the fast manual frequency hopping function of Rx is required in the applications, please refer to the method described in *AN197-CMT2300A-CMT2119B Fast Manual Frequency Hopping*. According to the *CMT2300A-CMT2219B Frequency Hopping Calculation Table*, fill the value of initial AFC_OVF_TH<7 The value of :0> into the 0x27 CUS_FSK4 register. If only Tx frequency hopping is required or frequency hopping is not required, users can skip this step.
8. Set the registers in control area 1 (0x60 – 0x6A) as required, set RSTN_IN_EN to 0, and set LOCKING_EN to 1.
9. Configure the 4th bit of the CUS_MODE_STA (0x61) register, CFG_RETAIN, to 1. This bit controls to make the value of the entire configuration area of 0x00 – 0x5F cannot be erased by a soft reset.
10. Send the go_sleep command, this action makes the register configuration take effect.

After the initialization, the chip is ready for operating. If partial or all registers contents change is required, switch to STBY first, then perform further configurations, after then switch to SLEEP to let the configurations take effect. This operation flow should be followed whenever register content change is required.

In addition, if the chip is operating in the fully automatic duty cycle mode, the MCU should make the chip exit duty cycle mode first then perform further configuration according to the above processing flow. See *AN164-CMT2219B Low Power Mode Usage Guide* for details of entering and exiting duty cycle mode . See Appendix 4 for the CMT2219B initialization code.

3.6 Function Division in Configuration Area

As mentioned above, the entire configuration space is divided into 7 functional areas corresponding to the divided areas in the register file generated by the RFPDK one by one. Users can change the corresponding parameter directly according to RFPDK

guidance with no need for understanding the related register details since the register contents are generated by RFPDK.

For example, if users need to modify data rate in the application. Configure data rate in RFPDK first, then export the entire register configuration content. In the exported file, pick up the content in data rate area and put it into an array then write the array into the MCU program. When changing data rate is required, write this part of content into the corresponding register address.

The following table lists the 7 area division in FSK and OOK modes respectively. They mainly differ in data rate area.

Table 8. Register Configuration Area Partition in FSK Mode for CMT2219B

Area	Address	Area in RFPDK Generated File
CMT area	0x00 – 0x0B	CMT bank
System area	0x0C – 0x17	System bank
Frequency area	0x18 – 0x1F	Frequency bank
Data Rate area	Require to fill in : 0x20 – 0x32	Data rate bank
	Con be ignored: 0x33– 0x37	
Baseband area	0x38 – 0x54	Baseband bank
Reserved area	0x55 – 0x5E	Reserved bank
LBD area	0x5F	LBD bank

Table 9. Register Configuration Area Partition in OOK Mode for CMT2219B

Area	Address	Area in RFPDK Generated File
CMT area	0x00 – 0x0B	CMT bank
System area	0x0C – 0x17	System bank
Frequency area	0x18 – 0x1F	Frequency bank
Data Rate area Baseband area	Require to fill in: 0x20 – 0x23	Data Rate bank
	Con be ignored: 0x24– 0x2A	
	Require to fill in: 0x2B – 0x37	
Reserve area	0x38 – 0x54	Baseband bank
LBD area	0x55 – 0x5E	Reserve bank
CMT area	0x5F	LBD bank

The negligible area in the table refers to the area that users do not need to configure. The description of each area is provided below. Among them, users do not need to understand the contents of the CMT area, frequency area and data rate area. Register contents in these area can be generated by RFPDK.

3.6.1 CMT Area (0x00 – 0x0B)

The CMT area is for CMOSTEK internal use. Users can use the register content generated by RFPDK directly.

Table 10. CMT Area

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	RW	CUS_CMT1	Import RFPDK generated configuration directly. Users do not need to understand them.							
0x01	RW	CUS_CMT2								
0x02	RW	CUS_CMT3								
0x03	RW	CUS_CMT4								
0x04	RW	CUS_CMT5								
0x05	RW	CUS_CMT6								
0x06	RW	CUS_CMT7								
0x07	RW	CUS_CMT8								
0x08	RW	CUS_CMT9								
0x09	RW	CUS_CMT10								
0x0A	RW	CUS_CMT11								
0x0B	RW	CUS_RSSI								

3.6.2 System Area (0x0C – 0x17)

Table 11. System Area

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0C	RW	CUS_SYS1	LMT_VTR [1:0]		MIXER_BIAS [1:0]		LNA_MODE [1:0]		LNA_BIAS [1:0]	
0x0D	RW	CUS_SYS2	LFOSC_RECAL_EN	LFOSC_CAL1_EN	LFOSC_CAL2_EN	RX_TIMER_EN	SLEEP_TIMER_EN	RESV	RX_DC_EN	DC_PAUSE
0x0E	RW	CUS_SYS3	SLEEP_BYPASS_EN	XTAL_STB_TIME [2:0]			RESV [1:0]		RX_EXIT_STATE [1:0]	
0x0F	RW	CUS_SYS4	SLEEP_TIMER_M [7:0]							
0x10	RW	CUS_SYS5	SLEEP_TIMER_M [10:8]				SLEEP_TIMER_R [3:0]			
0x11	RW	CUS_SYS6	RX_TIMER_T1_M [7:0]							
0x12	RW	CUS_SYS7	RX_TIMER_T1_M [10:8]				RX_TIMER_T1_R [3:0]			
0x13	RW	CUS_SYS8	RX_TIMER_T2_M [7:0]							
0x14	RW	CUS_SYS9	RX_TIMER_T2_M [10:8]				RX_TIMER_T2_R [3:0]			
0x15	RW	CUS_SYS10	COL_DET_EN	COL_OFS_SEL	RX_AUTO_EXIT_DIS	DOUT_MUTE	RX_EXTEND_MODE [3:0]			
0x16	RW	CUS_SYS11	PJD_TH_SEL	CCA_INT_SEL [1:0]		RSSI_DET_SEL [1:0]		RSSI_AVG_MODE [2:0]		
0x17	RW	CUS_SYS12	PJD_WIN_SEL [1:0]		CLKOUT_EN	CLKOUT_DIV [4:0]				

The parameters in system area control the system operation mode and some system features such as SLEEP TIMER enabling, DUTY CYCLE enabling, the method selection to implement ultra-low power reception using RSSI and PJD (phase jump detection).

3.6.3 Frequency Area (0x18 - 0x1F)

The frequency area is responsible for RX frequency points configuring. Users can use the register content generated by RFPDK directly. If RX frequency point change is required during the application operating period, the below 2 methods can be applied.

1. Use RFPDK to generate parameters corresponding to various frequency points. When it needs to change frequency points, rewrite all registers in frequency area.
2. Switch to a new frequency point using new manual frequency hopping mechanism based on the frequency point set in initialization phase.

Table 12. Frequency Area

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x18	RW	CUS_RF1	Import RFPDK generated configuration directly. Users do not need to understand them.							
0x19	RW	CUS_RF2								
0x1A	RW	CUS_RF3								
0x1B	RW	CUS_RF4								
0x1C	RW	CUS_RF5								
0x1D	RW	CUS_RF6								
0x1E	RW	CUS_RF7								
0x1F	RW	CUS_RF8								

3.6.4 Data Rate Area (0x20 – 0x37)

As mentioned above, the data rate area address partitioning depends on FSK and OOK. The negligible part does not need user to fill in.

As data rate affects a large number of parameters such as FSK or OOK demodulation parameters, clock recovery configurations and AGC configurations, the data rate area contains much content and covers relatively larger address range. When data rate change is required, users need prepare all data rate area parameters generated by RFPDK in advance then write them all in data rate area (excluding the negligible area).

Table 13. Data Rate Area

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x20	RW	CUS_RF9	Import RFPDK generated configuration directly. Users do not need to understand them.							
0x21	RW	CUS_RF10								
0x22	RW	CUS_RF11								
0x23	RW	CUS_RF12								
0x24	RW	CUS_FSK1								
0x25	RW	CUS_FSK2								
0x26	RW	CUS_FSK3								
0x27	RW	CUS_FSK4								
0x28	RW	CUS_FSK5								
0x29	RW	CUS_FSK6								
0x2A	RW	CUS_FSK7								
0x2B	RW	CUS_CDR1								
0x2C	RW	CUS_CDR2								
0x2D	RW	CUS_CDR3								
0x2E	RW	CUS_CDR4								
0x2F	RW	CUS_AGC1								
0x30	RW	CUS_AGC2								
0x31	RW	CUS_AGC3								
0x32	RW	CUS_AGC4								
0x33	RW	CUS_OOK1								
0x34	RW	CUS_OOK2								
0x35	RW	CUS_OOK3								
0x36	RW	CUS_OOK4								
0x37	RW	CUS_OOK5								

3.6.5 Baseband Area (0x38 - 0x54)

Table 14. Baseband Area

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x38	RW	CUS_PKT1	RX_PREAM_SIZE [4:0]					PREAM_LENG_UNIT	DATA_MODE [1:0]	
0x39	RW	CUS_PKT2	RESV [7:0]							
0x3A	RW	CUS_PKT3	RESV [7:0]							
0x3B	RW	CUS_PKT4	PREAM_VALUE [7:0]							
0x3C	RW	CUS_PKT5	RESV	SYNC_TOL [2:0]			SYNC_SIZE [2:0]			SYNC_MAN_EN
0x3D	RW	CUS_PKT6	SYNC_VALUE [7:0]							
0x3E	RW	CUS_PKT7	SYNC_VALUE [15:8]							
0x3F	RW	CUS_PKT8	SYNC_VALUE [23:16]							
0x40	RW	CUS_PKT9	SYNC_VALUE [31:24]							
0x41	RW	CUS_PKT10	SYNC_VALUE [39:32]							
0x42	RW	CUS_PKT11	SYNC_VALUE [47:40]							
0x43	RW	CUS_PKT12	SYNC_VALUE [55:48]							
0x44	RW	CUS_PKT13	SYNC_VALUE [63:56]							
0x45	RW	CUS_PKT14	RESV	PAYLOAD_LENG [10:8]			AUTO_ACK_EN	NODE_LENG_POS_SEL	PAYLOAD_BIT_ORDER	PKT_TYPE
0x46	RW	CUS_PKT15	PAYLOAD_LENG [7:0]							
0x47	RW	CUS_PKT16	RESV	RESV	NODE_FREE_EN	NODE_ERR_MASK	NODE_SIZE [1:0]		NODE_DET_MODE [1:0]	
0x48	RW	CUS_PKT17	NODE_VALUE [7:0]							
0x49	RW	CUS_PKT18	NODE_VALUE [15:8]							
0x4A	RW	CUS_PKT19	NODE_VALUE [23:16]							
0x4B	RW	CUS_PKT20	NODE_VALUE [31:24]							
0x4C	RW	CUS_PKT21	FEC_TYPE	FEC_EN	CRC_BYTE_SWAP	CRC_BIT_INV	CRC_RANGE	CRC_TYPE [1:0]		CRC_EN
0x4D	RW	CUS_PKT22	CRC_SEED [7:0]							
0x4E	RW	CUS_PKT23	CRC_SEED [15:8]							
0x4F	RW	CUS_PKT24	CRC_BIT_ORDER	WHITEN_SEED [8]	WHITEN_SEED_TYPE	WHITEN_TYPE [1:0]		WHITEN_EN	MANCH_TYPE	MANCH_EN
0x50	RW	CUS_PKT25	WHITEN_SEED [7:0]							
0x51	RW	CUS_PKT26	RESV [7:0]							
0x52	RW	CUS_PKT27	RESV [7:0]							
0x53	RW	CUS_PKT28	RESV [7:0]							
0x54	RW	CUS_PKT29	FIFO_AUTO_RES_EN	FIFO_TH [6:0]						

The baseband area majorly covers configurations of received data format, encoding & decoding and partial of FIFO configurations (registers for FIFO usage & control are mainly located in control area). The content in this part corresponds to RFPDK parameters one by one. Users need to make clear the parameter meanings before configuration.

3.6.6 Reserved Area（0x55 – 0x5E）

Table 15. Baseband Area

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x53	RW	CUS_RESV1	Reserved area, users do not need to write in							
0x54	RW	CUS_RESV1								
0x57	RW	CUS_RESV1								
0x58	RW	CUS_RESV1								
0x59	RW	CUS_RESV1								
0x5A	RW	CUS_RESV1								
0x5B	RW	CUS_RESV1								
0x5C	RW	CUS_RESV1								
0x5D	RW	CUS_RESV1								
0x5E	RW	CUS_RESV1								

User can ignore this area. Users can write all 0 into the area or do not write it.

3.6.7 LBD Area (0x5F)

Table 16. LBD Area

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x5F	RW	CUS_LBD	LBD_TH [7:0]							

This area covers the configuration of the LBD (low battery detection) comparison threshold only.

3.7 Control Area

As mentioned above, control area consists of control area 1 and control area 2. Control area 1 can be saved in SLEEP state, however control area 2 cannot be saved in SLEEP state.

- **Control Area 1 (0x60 - 0x6A)**

Table 17. Control Area 1

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x60	RW	CUS_MODE_CTL	CHIP_MODE_SWT [7:0]							
0x61	RW	CUS_MODE_STA	RESV [1:0]		RSTN_IN_EN	CFG_RETAIN	CHIP_MODE_STA [3:0]			
0x62	RW	CUS_EN_CTL	RESV [1:0]		RESV	FIFO_AUTO_CLR_DIS	RESV [3:0]			
0x63	RW	CUS_FREQ_CHNL	FH_CHANNEL [7:0]							
0x64	RW	CUS_FREQ_OFS	FH_OFFSET [7:0]							
0x65	RW	CUS_IO_SEL	RESV [1:0]		GPIO3_SEL [1:0]		GPIO2_SEL [1:0]		GPIO1_SEL [1:0]	
0x66	RW	CUS_INT1_CTL	RESV [1:0]		INT_POLAR	INT1_SEL [4:0]				
0x67	RW	CUS_INT2_CTL	RESV	LFOSC_OUT_EN	RESV	INT2_SEL [4:0]				
0x68	RW	CUS_INT_EN	SL_TMO_EN	RX_TMO_EN	RESV	PREAM_OK_EN	SYNC_OK_EN	NODE_OK_EN	CRC_OK_EN	PKT_DONE_EN
0x69	RW	CUS_FIFO_CTL	RESV [2:0]			FIFO_AUTO_CLR_DIS	RESV [1:0]		FIFO_MERGE_EN	RESV
0x6A	W	CUS_INT_CLR1	RESV [1:0]		SL_TMO_FLG	RX_TMO_FLG	RESV [1:0]		SL_TMO_CLR	RX_TMO_CLR

- **Control Area 2 (0x6B – 0x71)**

Table 18. Control Area 2

Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x6B	W	CUS_INT_CLR2	RESV [1:0]		LBD_CLR	PREAM_OK_CLR	SYNC_OK_CLR	NODE_OK_CLR	CRC_OK_CLR	PKT_DONE_CLR
0x6C	W	CUS_FIFO_CLR	RESV [4:0]					FIFO_RESTORE	FIFO_CLR_RX	RESV
0x6D	R	CUS_INT_FLAG	LBD_FLG	COL_ERR_FLG	PKT_ERR_FLG	PREAM_OK_FLG	SYNC_OK_FLG	NODE_OK_FLG	CRC_OK_FLG	PKT_OK_FLG
0x6E	R	CUS_FIFO_FLAG	RESV	RX_FIFO_FULL_FLG	RX_FIFO_NMTY_FLG	RX_FIFO_TH_FLG	RX_FIFO_OVF_FLG	RESV [2:0]		
0x6F	R	CUS_RSSI_CODE	RSSI_CODE [7:0]							
0x70	R	CUS_RSSI_DBM	RSSI_DBM [7:0]							
0x71	R	CUS_LBD_RESULT	LBD_RESULT [7:0]							

By operating control area, users can fulfill chip operating mode switching, IO and interrupt control, FIFO control, manual frequency hopping, RSSI reading, etc. Therefore, users may operate control area quite frequently. It should be noted that when writing the bits marked as RESV (grey color in above figures), users can only write in all 0 or do not have writing operation on them.

3.8 Operation Flow

As a summary, users operate the chip following the 3 steps below.

1. Read the related AN document. Generate a desired register file using RFPDK.
2. Perform initialization configuration process on the chip.
3. Run the application. During the application operating period, it includes 2 kinds of operations.

   a) Make configuration area changes

   Figure out the corresponding configuration area for the desired configuration. Use RFPDK to generate a new configuration table. Pick up the area that needs change and import them into the program. Write the content into the corresponding registers based on specific application requirements.

   b) Make control on the chip

   Understand the function of control area 1 and control area 2. Read AN document carefully to make clear the chip features and operating rules including FIFO control, IO and interrupt control, RSSI reading, manual Frequency hopping, and so on. Then operate the chip to fulfill application requirements.

4 CMT2219B_DemoEasy Introduction

This chapter discusses the demo program for the CMT2219B. The demo program is closely associated with thr various operations mentioned above. The source code of the functions called in the demo program is be provided in chapter 5.

4.1 Software Structure

To standardize the CMT2219B operation process and enhance software portability, the demo program is layered with each module calling corresponding API functions. The entire program is divided into the following 5 layers.

1. Application, which is application layer. It simply performs packets sending and receiving in the demo.
2. Radio handlers, which is chip processing layer, including chip initialization, configuration, state control and other processes.
3. CMT2219Bdrivers, which is chip driver layer. It is available for upper layer calling.
4. Hardware abstraction layer, which implements the access and control of chip register, FIFO and GPIO.
5. Hardware, which is hardware layer, including resources provided by MCU such as LED, button and SPI communication.

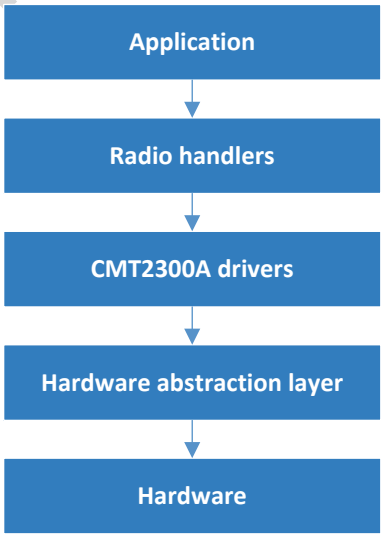


Figure 8. Software Layer Diagram

4.2 Software Implementation and Calling Relationship

This chapter discusses calling relationships between modules, initialization flow and operating flow. The below 5 major files are included.

1. main.c： application implementation
2. radio.c： chip processing layer implementation
3. cmt2219b.c： chip diver layer implementation
4. cmt2219b_hal.c： abstract hardware layer implementation
5. cmt_spi3.c： chip SPI analog timing implementation including register and FIFO access timing.

The first 4 layers and the functions of each layer are shown in the below figure. Among them, the file cmt_sp3.c provides SPI functions.

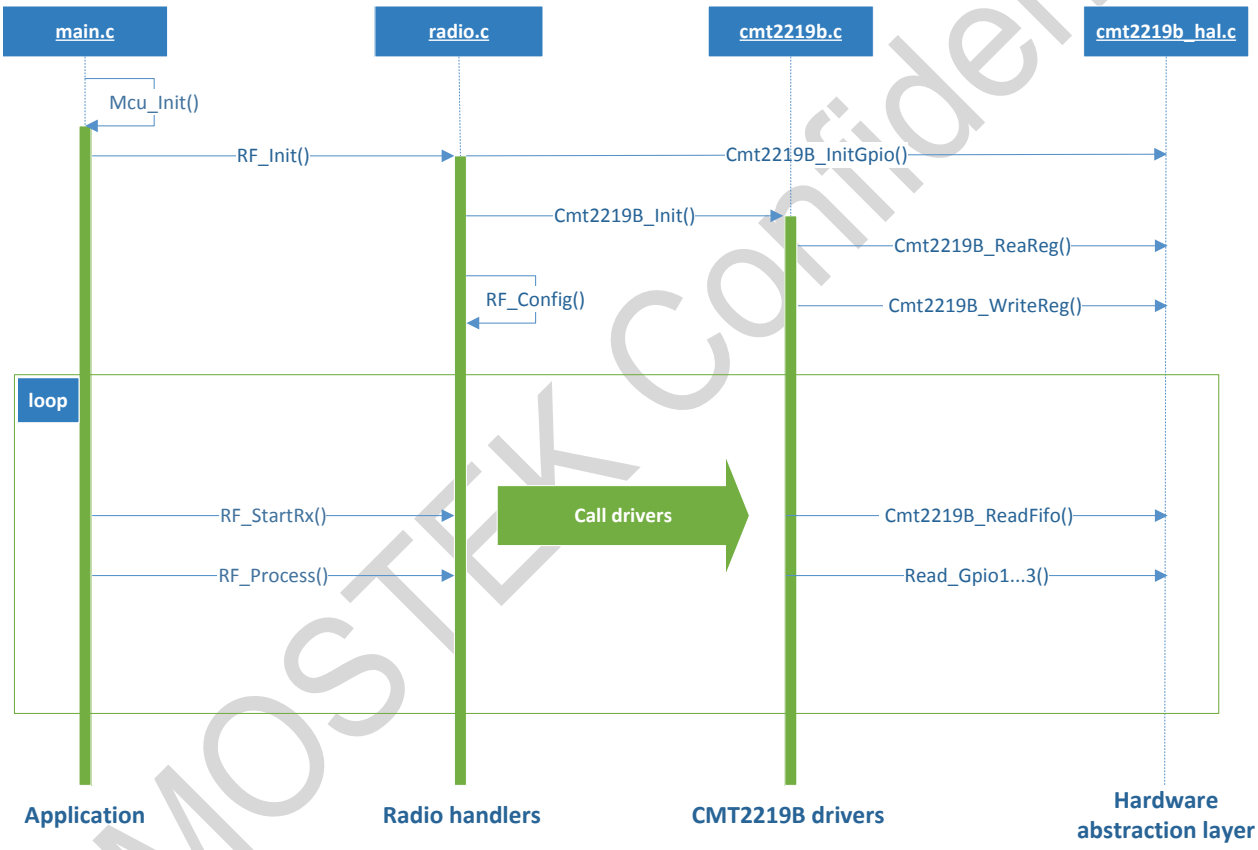


Figure 9. Software Implementation and Calling Relationship

4.2.1 CMT2219B Initialization

After chip power up, the RF_Init() function must be called for initialization, which includes the initialization of SPI and GPIO1/2/3, soft reset, as well as enabling and disabling of some registers. The initialization flow chart is shown in the below figure.

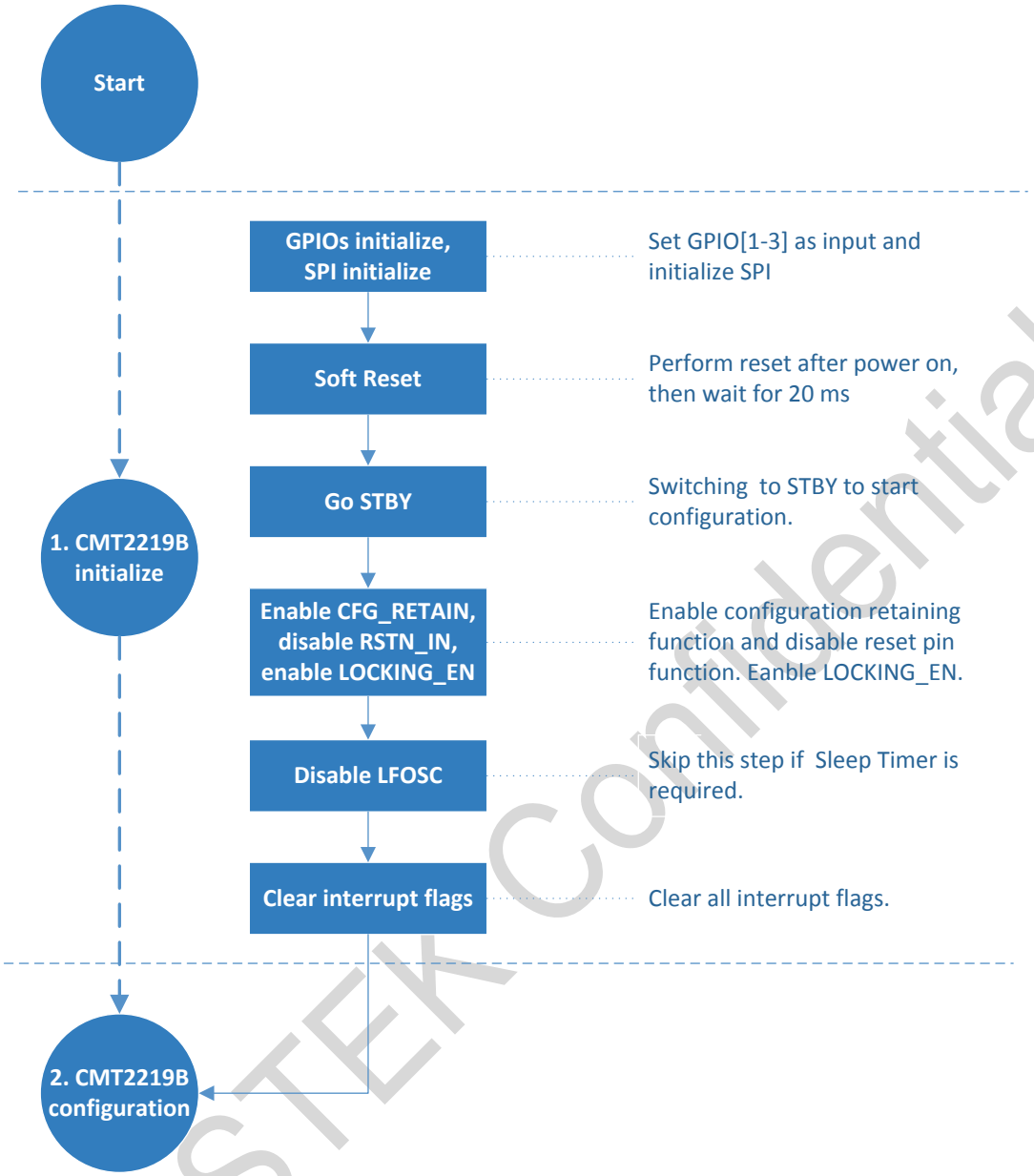


Figure 10. CMT2219B Initialization Flow Chart

4.2.2 CMT2219B Configuration

After initialization, the function RF_Config should be called. It configures chip registers, interrupts, and GPIOs as well as enters SLEEP finally to make the configurations come into effect. The configuration flow chart is shown in the below figure.

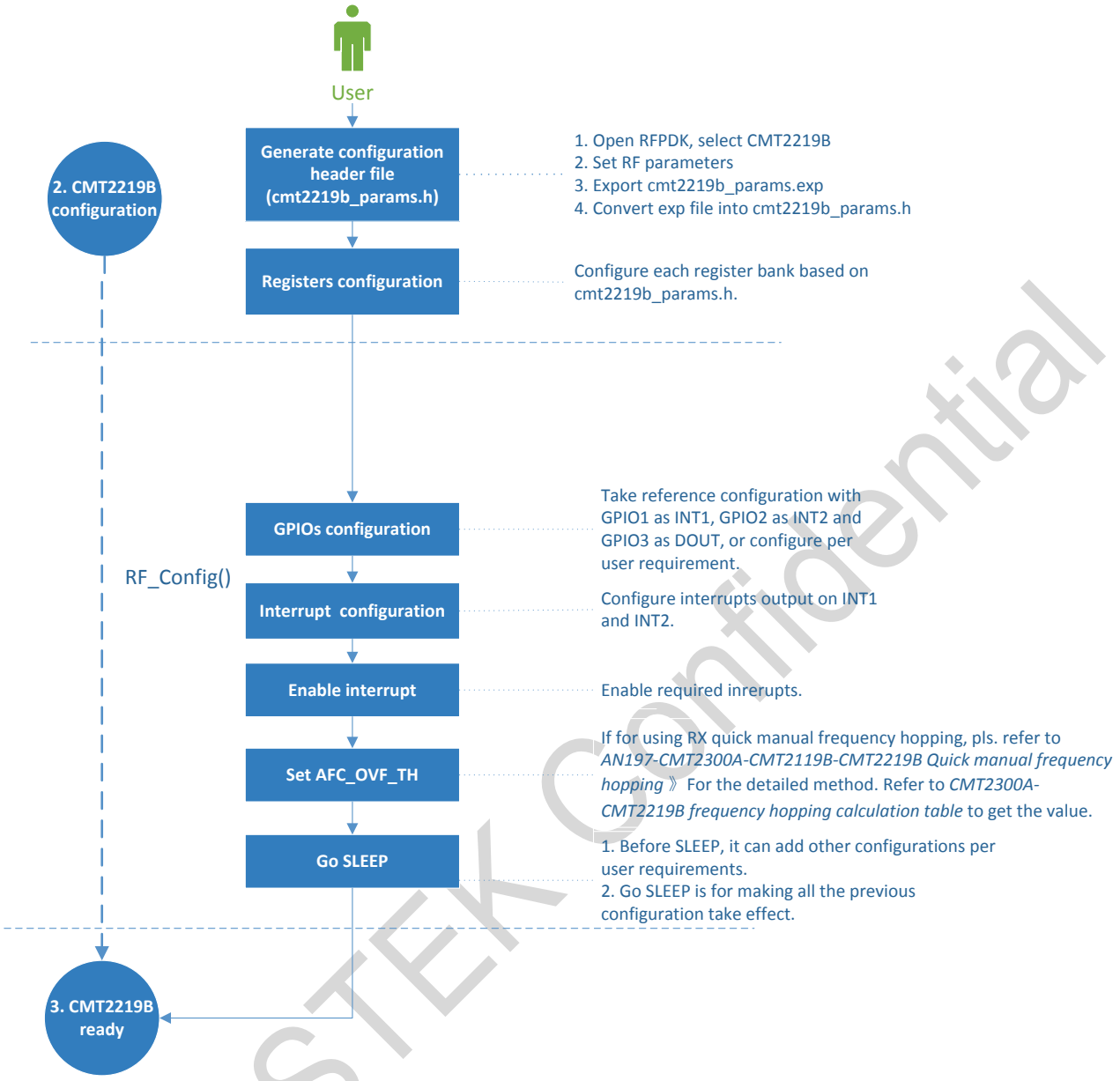


Figure 11. CMT2219B Configuration Flow Chart

4.2.3 CMT2219B State Handling

After configuration, the function RF_StartRx() can be called to enter Rx state. Then the RF_Process() function is called cyclically for communication processing. The RF_Process() uses a state machine to control the chip and return state results to the application layer for further processing.

1. RF_STATE_IDLE： Idle state
2. RF_STATE_RX_START： Rx start state. FIFO read is enabled for receiving.
3. RF_STATE_RX_WAIT： Rx waiting state. It detects complete reception interrupt continuously
4. RF_STATE_RX_DONE： Rx done state. It reads FIFO, check and clear interrupt
5. RF_STATE_RX_TIMEOUT： Rx timeout state. It makes the chip exit receiving.
6. RF_STATE_ERROR： Error state. It cannot enter Rx and needs reset the chip and perform configuration again.

The return results of RF_Process() are as follows.

1. RF_IDLE：　　Chip idle. It can enter Rx.
2. RF_BUSY：　　Chip busy. It is in data receiving.
3. RF_RX_DONE：　　Chip Rx done. The upper layer can perform receiving handling.
4. RF_RX_TIMEOUT：　　Chip Rx timeout.
5. RF_ERROR：　　Chip Rx error.

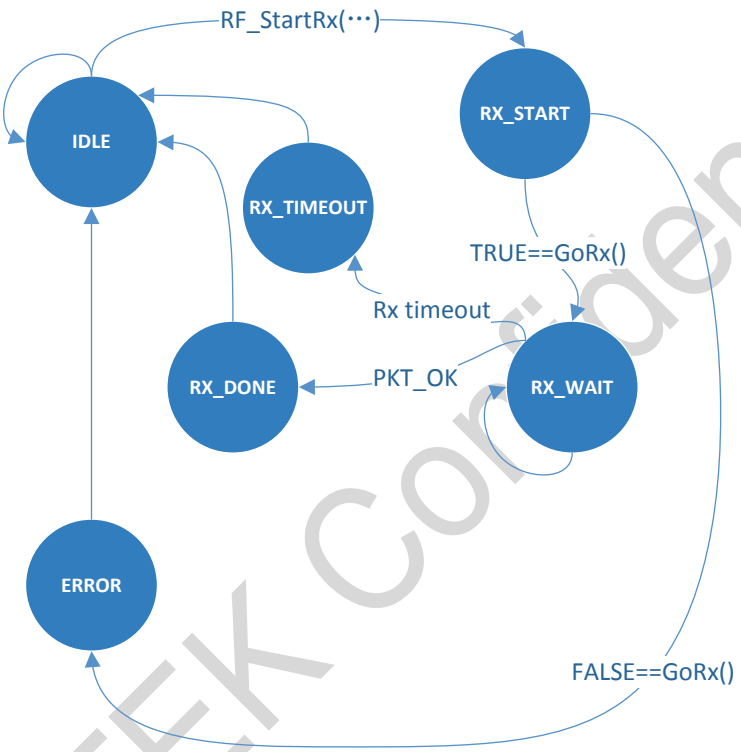


Figure 12. CMT2219B State Processing Diagram

4.3 Software Directory Structure

The demo program is based on RFEB platform and developed based on Keil5 IDE, which has a complete project.

1. Libraries：　　STM32F103 related library files
2. MDK-ARM：　　Keil5 related project and compiling files
3. USER：　　Demo source program
4. clear.bat：　　Execution file for clearing all intermediate files.
5. services：　　Time, interrupt, etc. services implemented based on MCU
6. platform：　　Platform related configuration or control files
7. periph：　　Peripheral resources such as LED, key, LCD and SPI
8. radio：　　All API interface files for CMT2219B
9. cmt2219b_params.h:　　The header file for RFPDK exp file conversion, corresponding to the register banks of the exp file one by one.
10. cmt2219b_defs.h：　　Register address macro definition for CMT2219B chip

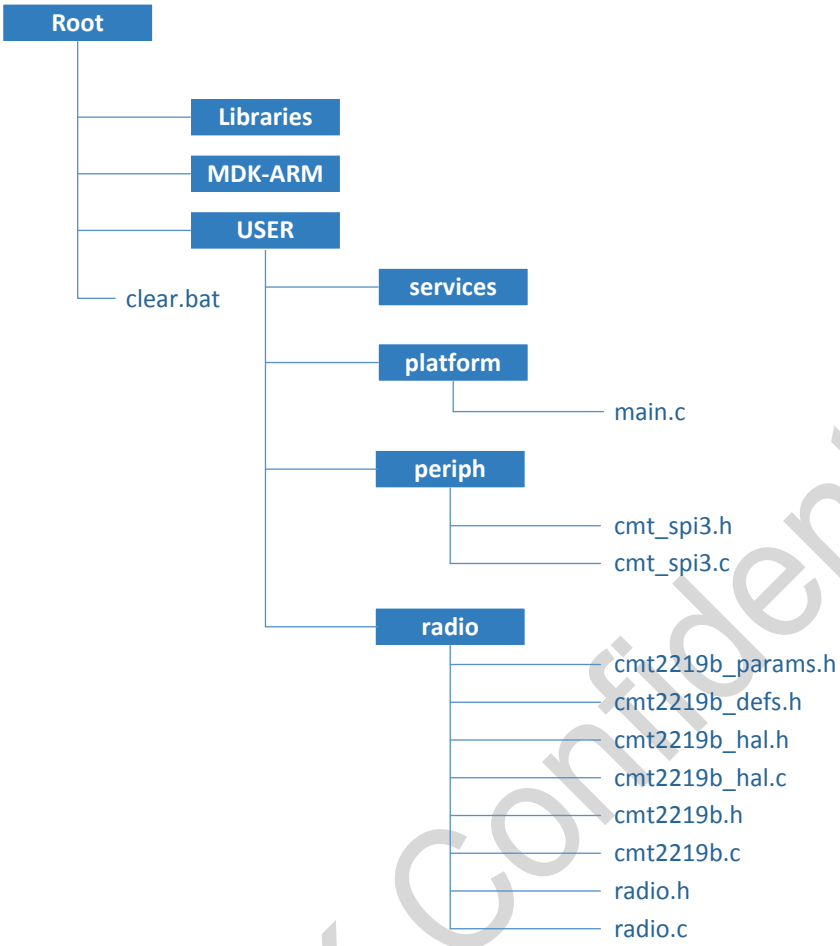


Figure 13. Software Directory Structure

4.3.1 Application Layer Source Code

Main.c is application layer source code calling OnReceive() for the CMT2219B receiving communication.

```
#define RF_PACKET_SIZE 32              /* Define the payload size here */

static u8 g_rxBuffer[RF_PACKET_SIZE];   /* RF Rx buffer */

static u16 g_nRecvCount = 0;
static u16 g_nErrCount  = 0;

void Mcu_Init(void);

/* Manages the receiver operation */
void OnReceive(void);
```

4.3.2 Analog SPI Implementation Source Code

cmt_spi3.c is the SPI communication implementation for CMT2219B. If for migration to other MCU platforms, the following macro definition should be modified.

```
/* ****************************************************************************
*  The following need to be modified by user
*  ************************************************************************** */
#define cmt_spi3_csb_out()      SET_GPIO_OUT(CMT_CSB_GPIO)
#define cmt_spi3_fcsb_out()     SET_GPIO_OUT(CMT_FCSB_GPIO)
#define cmt_spi3_sclk_out()     SET_GPIO_OUT(CMT_SCLK_GPIO)
#define cmt_spi3_sdio_out()     SET_GPIO_OUT(CMT_SDIO_GPIO)
#define cmt_spi3_sdio_in()      SET_GPIO_IN(CMT_SDIO_GPIO)

#define cmt_spi3_csb_1()        SET_GPIO_H(CMT_CSB_GPIO)
#define cmt_spi3_csb_0()        SET_GPIO_L(CMT_CSB_GPIO)

#define cmt_spi3_fcsb_1()       SET_GPIO_H(CMT_FCSB_GPIO)
#define cmt_spi3_fcsb_0()       SET_GPIO_L(CMT_FCSB_GPIO)

#define cmt_spi3_sclk_1()       SET_GPIO_H(CMT_SCLK_GPIO)
#define cmt_spi3_sclk_0()       SET_GPIO_L(CMT_SCLK_GPIO)

#define cmt_spi3_sdio_1()       SET_GPIO_H(CMT_SDIO_GPIO)
#define cmt_spi3_sdio_0()       SET_GPIO_L(CMT_SDIO_GPIO)
#define cmt_spi3_sdio_read()    READ_GPIO_PIN(CMT_SDIO_GPIO)
/* ****************************************************************************** */
```

4.3.3 Subtract Hardware Layer Source Code

cmt2219b_hal.c is the source code for subtract hardware later source code implementing register, FIFO and GPIO access interfaces for CMT2219B.

```
/*! ****************************************************
* @name    CMT2219B_InitGpio
* @desc    Initializes the CMT2219B interface GPIOs.
* ****************************************************/
void CMT2219B_InitGpio(void);

/*! ****************************************************
* @name    CMT2219B_ReadReg
* @desc    Read the CMT2219B register at the specified address.
* @paramaddr: register address
* @return  Register value
* ****************************************************/
u8 CMT2219B_ReadReg(u8 addr);
```

```
/*! *********************************************************
* @name    CMT2219B_WriteReg
* @desc    Write the CMT2219B register at the specified address.
* @paramaddr: register address
*         dat: register value
* *********************************************************/
void CMT2219B_WriteReg(u8 addr, u8 dat);

/*! *********************************************************
* @name    CMT2219B_ReadFifo
* @desc    Reads the contents of the CMT2219B FIFO.
* @parambuf: buffer where to copy the FIFO read data
*         len: number of bytes to be read from the FIFO
* *********************************************************/
void CMT2219B_ReadFifo(u8 buf[], u16 len);

/*! *********************************************************
* @name    CMT2219B_WriteFifo
* @desc    Writes the buffer contents to the CMT2219B FIFO.
* @parambuf: buffer containing data to be put on the FIFO
*         len: number of bytes to be written to the FIFO
* *********************************************************/
void CMT2219B_WriteFifo(const u8 buf[], u16 len);
```

If for demo program migration to other MCU platforms, some macro definition in cmt2219b_hal.h should be modified.

```
/* ***************************************************************************
*  The following need to be modified by user
*  ************************************************************************* */
#define CMT2219B_SetGpio1In()          SET_GPIO_IN(CMT_GPIO1_GPIO)
#define CMT2219B_SetGpio2In()          SET_GPIO_IN(CMT_GPIO2_GPIO)
#define CMT2219B_SetGpio3In()          SET_GPIO_IN(CMT_GPIO3_GPIO)
#define CMT2219B_ReadGpio1()           READ_GPIO_PIN(CMT_GPIO1_GPIO)
#define CMT2219B_ReadGpio2()           READ_GPIO_PIN(CMT_GPIO2_GPIO)
#define CMT2219B_ReadGpio3()           READ_GPIO_PIN(CMT_GPIO3_GPIO)
#define CMT2219B_DelayMs(ms)           system_delay_ms(ms)
#define CMT2219B_DelayUs(us)           system_delay_us(us)
#define CMT2219B_GetTickCount()        g_nSysTickCount
/* ************************************************************************* */
```

4.3.4 Chip Driver Layer Source Code

Cmt2219b.c is chip driver layer source code, implementing state switching operation, interrupt operation, GPIO operation, FIFO operation, general purpose register configuration & access, etc. This part is firmware program which is MCU platform

independent. Users do not need modification on it.

```
/*! *********************************************************
* @name    CMT2219B_Init
* @desc    Initialize chip status.
* *********************************************************/
void CMT2219B_Init(void)
{
    u8 tmp;

    CMT2219B_SoftReset();
    CMT2219B_DelayMs(20);

    CMT2219B_GoStby();

tmp  = CMT2219B_ReadReg(CMT2219B_CUS_MODE_STA);
tmp |= CMT2219B_MASK_CFG_RETAIN;        /* Enable CFG_RETAIN */
tmp&= ~CMT2219B_MASK_RSTN_IN_EN;       /* Disable RSTN_IN */
    CMT2219B_WriteReg(CMT2219B_CUS_MODE_STA, tmp);

tmp  = CMT2219B_ReadReg(CMT2219B_CUS_EN_CTL);
tmp |= CMT2219B_MASK_LOCKING_EN;        /* Enable LOCKING_EN */
    CMT2219B_WriteReg(CMT2219B_CUS_EN_CTL, tmp);

    CMT2219B_EnableLfosc(FALSE);            /* Diable LFOSC */

    CMT2219B_ClearInterruptFlags();
}
```

4.3.5 Chip Processing Layer Source Code

Radio.c is chip processing layer source code.

```
/* RF state machine */
typedef enum {
    RF_STATE_IDLE = 0,
    RF_STATE_RX_START,
    RF_STATE_RX_WAIT,
    RF_STATE_RX_DONE,
    RF_STATE_RX_TIMEOUT,
    RF_STATE_ERROR,
} EnumRFStatus;

/* RF process function results */
typedef enum {
    RF_IDLE = 0,
```

```
    RF_BUSY,
    RF_RX_DONE,
    RF_RX_TIMEOUT,
    RF_ERROR,
} EnumRFResult;
```

5 Appendix

5.1 Appendix 1: Sample Code for SPI Read & Write Operation

```
void cmt_spi3_write(u8 addr, u8 dat)
{
    cmt_spi3_sdio_1();
    cmt_spi3_sdio_out();

    cmt_spi3_sclk_0();
    cmt_spi3_sclk_out();
    cmt_spi3_sclk_0();

    cmt_spi3_fcsb_1();
    cmt_spi3_fcsb_out();
    cmt_spi3_fcsb_1();

    cmt_spi3_csb_0();

    /* > 0.5 SCLK cycle */
    cmt_spi3_delay();
    cmt_spi3_delay();

    /* r/w = 0 */
    cmt_spi3_send(addr&0x7F);

    cmt_spi3_send(dat);

    cmt_spi3_sclk_0();

    /* > 0.5 SCLK cycle */
    cmt_spi3_delay();
    cmt_spi3_delay();

    cmt_spi3_csb_1();
```

```
    cmt_spi3_sdio_1();
    cmt_spi3_sdio_in();

    cmt_spi3_fcsb_1();
}

void cmt_spi3_read(u8 addr, u8* p_dat)
{
    cmt_spi3_sdio_1();
    cmt_spi3_sdio_out();

    cmt_spi3_sclk_0();
    cmt_spi3_sclk_out();
    cmt_spi3_sclk_0();

    cmt_spi3_fcsb_1();
    cmt_spi3_fcsb_out();
    cmt_spi3_fcsb_1();

    cmt_spi3_csb_0();

    /* > 0.5 SCLK cycle */
    cmt_spi3_delay();
    cmt_spi3_delay();

    /* r/w = 1 */
    cmt_spi3_send(addr|0x80);

    /* Must set SDIO to input before the falling edge of SCLK */
    cmt_spi3_sdio_in();

    *p_dat = cmt_spi3_recv();

    cmt_spi3_sclk_0();

    /* > 0.5 SCLK cycle */
    cmt_spi3_delay();
    cmt_spi3_delay();

    cmt_spi3_csb_1();

    cmt_spi3_sdio_1();
    cmt_spi3_sdio_in();

    cmt_spi3_fcsb_1();
```

```
}
```

5.2 Appendix 2: Sample Code for SPI Reading FIFO Operation

The example code for SPI reading FIFO sub-function.

```
void cmt_spi3_read_fifo(u8* p_buf, u16 len)
{
    u16 i;

    cmt_spi3_fcsb_1();
    cmt_spi3_fcsb_out();
    cmt_spi3_fcsb_1();

    cmt_spi3_csb_1();
    cmt_spi3_csb_out();
    cmt_spi3_csb_1();

    cmt_spi3_sclk_0();
    cmt_spi3_sclk_out();
    cmt_spi3_sclk_0();

    cmt_spi3_sdio_in();

for(i=0; i<len; i++)
    {
        cmt_spi3_fcsb_0();

        /* > 1 SCLK cycle */
        cmt_spi3_delay();
        cmt_spi3_delay();

p_buf[i] = cmt_spi3_recv();

        cmt_spi3_sclk_0();

        /* > 2 us */
        cmt_spi3_delay_us();
        cmt_spi3_delay_us();
        cmt_spi3_delay_us();

        cmt_spi3_fcsb_1();
```

```
        /* > 4 us */
        cmt_spi3_delay_us();
        cmt_spi3_delay_us();
        cmt_spi3_delay_us();
        cmt_spi3_delay_us();
        cmt_spi3_delay_us();
        cmt_spi3_delay_us();
    }

    cmt_spi3_sdio_in();

    cmt_spi3_fcsb_1();
}
```

5.3 Appendix 3: Sample Code for State Switching Library Function

```
/*! ********************************************************
* @name    CMT2219B_GoSleep
* @desc    Entry SLEEP mode.
* @return  TRUE or FALSE
* ********************************************************/
BOOL CMT2219B_GoSleep(void)
{
    return CMT2219B_AutoSwitchStatus(CMT2219B_GO_SLEEP);
}

/*! ********************************************************
* @name    CMT2219B_GoStby
* @desc    Entry Sleep mode.
* @return  TRUE or FALSE
* ********************************************************/
BOOL CMT2219B_GoStby(void)
{
    return CMT2219B_AutoSwitchStatus(CMT2219B_GO_STBY);
}

/*! ********************************************************
* @name    CMT2219B_GoRFS
* @desc    Entry RFS mode.
* @return  TRUE or FALSE
* ********************************************************/
BOOL CMT2219B_GoRFS(void)
{
    return CMT2219B_AutoSwitchStatus(CMT2219B_GO_RFS);
}
```

```
/*! *********************************************************
* @name    CMT2219B_GoRx
* @desc    Entry Rx mode.
* @return  TRUE or FALSE
* *********************************************************/
BOOL CMT2219B_GoRx(void)
{
    return CMT2219B_AutoSwitchStatus(CMT2219B_GO_RX);
}
```

5.4 Appendix 4: Sample Code for Initialization Function

```
void RF_Init(void)
{
    u8 tmp;

    CMT2219B_InitGpio();
     CMT2219B_Init();

    /* Config registers */
    CMT2219B_ConfigRegBank(CMT2219B_CMT_BANK_ADDR        , g_cmt2219bCmtBank          ,
CMT2219B_CMT_BANK_SIZE      );
    CMT2219B_ConfigRegBank(CMT2219B_SYSTEM_BANK_ADDR      , g_cmt2219bSystemBank       ,
CMT2219B_SYSTEM_BANK_SIZE    );
    CMT2219B_ConfigRegBank(CMT2219B_FREQUENCY_BANK_ADDR   ,  g_cmt2219bFrequencyBank   ,
CMT2219B_FREQUENCY_BANK_SIZE );
    CMT2219B_ConfigRegBank(CMT2219B_DATA_RATE_BANK_ADDR   , g_cmt2219bDataRateBank     ,
CMT2219B_DATA_RATE_BANK_SIZE );
    CMT2219B_ConfigRegBank(CMT2219B_BASEBAND_BANK_ADDR    ,  g_cmt2219bBasebandBank    ,
CMT2219B_BASEBAND_BANK_SIZE );
    CMT2219B_ConfigRegBank(CMT2219B_RESERVE_BANK_ADDR     , g_cmt2219bReserveBank      ,
CMT2219B_RESERVE_BANK_SIZE   );
    CMT2219B_ConfigRegBank(CMT2219B_LBD_BANK_ADDR         , g_cmt2219bLbdBank          ,
CMT2219B_LBD_BANK_SIZE      );

    // xosc_aac_code[2:0] = 2
tmp = (~0x07) & CMT2219B_ReadReg(CMT2219B_CUS_CMT10);
    CMT2219B_WriteReg(CMT2219B_CUS_CMT10, tmp|0x02);

     RF_Config();
}
```

6 Revise History

Table 19. Revise History Records

Version No.	Chapter	Description	Date
1.0	All	Initial version	2017-11-21

7 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: 518057
Sales: sales@cmostek.com
Supports: support@cmostek.com
Website: www.cmostek.com