CMOSTEK AN167

CMT2219B FIFO and Packet Format Operation Guide

Overview

This document discusses how the CMT2219B supports FIFO, packet format and interrupt system. When discussing register configuration, the document describes the corresponding input parameters in RFPDK as well to guide users' configuration.

The product models covered in this document are listed 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	Receiving	EEPROM	QFN16

Before reading this document, it is recommended to read the *AN161-CMT2219B Quick Start Guide* to understand the basic information of the product.

Table of Contents

1 How FIFO Works 3

1.1 FIFO Related Registers 3

1.2 FIFO Operating Mode 4

1.3 Interrupt Timing for FIFO 5

1.4 FIFO Usage 5

2 Packet Format 7

2.1 Data Mode Configuration 7

2.2 Preamble Configuration 8

2.3 Sync Word Configuration 9

2.4 Overall Data Packet Configuration 11

2.5 Node ID Configuration 13

2.6 FEC Configuration 17

2.7 CRC Configuration 18

2.8 Encoding and Decoding Configuration 21

3 GPIO and Interrupt 22

3.1 GPIO Configuration 22

3.2 Interrupt Configuration and Mapping 23

4 Appendix 27

Appendix 1. Sample Code for FIFO Read and Write Operation 27

Appendix 2. Sample Code for GPIO Output Interrupt Configuration Function 28

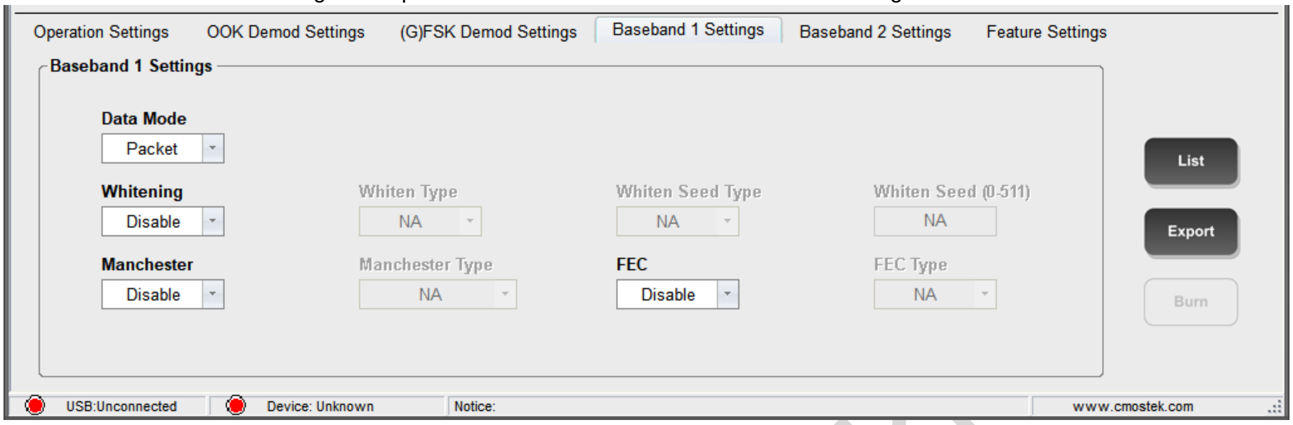
5 Revise History 30

6 Contacts 31

1 How FIFO Works

1.1 FIFO Related Registers

The FIFO related RFPDK configuration parameters and screen are shown in the below figure.



Baseband 1 Settings

Data Mode: Packet
Whitening: Disable
Whiten Type: NA
Whiten Seed Type: NA
Whiten Seed (0-511): NA
Manchester: Disable
Manchester Type: NA
FEC: Disable
FEC Type: NA

USB:Unconnected | Device: Unknown | Notice: | www.cmostek.com

Figure 1. FIFO Screen in RFPDK

Table 2. FIFO Related Parameters

RFPDK Parameter	Register Bit
Data Mode	DATA_MODE <1:0>
This parameter is not shown in RFPDK. It is configured flexibly in user application programs .	FIFO_TH <6:0>

The register descriptions are as follows.

Table 3. Registers in Configuration Area

Register	Bit	R/W	Flag	Description
CUS_PKT1 (0x38)	0	RW	DATA_MODE <1:0>	It determines data processing mode. 0: direct mode (default) 1: NA 2: packet mode 3: NA
CUS_PKT29 (0x54)	6:0	RW	FIFO_TH <6:0>	FIFO input threshold (unit: byte). Once unread data exceeds the threshold, RX_FIFO_TH_FLG is set to 1. When FIFO_MERGE_EN = 0, it ranges from 1 to 31. When FIFO_MERGE_EN = 1, it ranges from 1 to 63.

Table 4. Registers in Control Area 1

Register	Bit	R/W	Flag	Description
CUS_FIFO_CTL (0x69)	4	RW	FIFO_AUTO_CLR_DIS	0: clear FIFO to 0 before entering RX. 1: do not clear to 0 automatically
	1	RW	FIFO_MERGE_EN	0: the size of FIFO is 32 bytes. 1: the size of FIFO is 64 bytes.

Table 5. Registers in Control Area 2

Register	Bit	R/W	Flag	Description
CUS_FIFO_CLR (0x6C)	1	W	FIFO_CLR_RX	0: invalid. 1: clear FIFO. After setting it to 1, users do not need to set it back to 0 since it will be set to 0 internally.
CUS_FIFO_FLAG (0x6E)	6	R	RX_FIFO_FULL_FLG	The interrupt flag indicating FIFO is not empty. 0: invalid. 1: valid.
	5	R	RX_FIFO_NMTY_FLG	The interrupt flag indicating unread content exceeds FIFO TH. 0: invalid. 1: valid.
	4	R	RX_FIFO_TH_FLG	The interrupt flag indicating FIFO is full. 0: invalid. 1: valid.
	3	R	RX_FIFO_OVF_FLG	The interrupt flag indicating FIFO overflows. 0: invalid. 1: valid.
Notes: The register polarities are controlled by INT_POLAR, meaning that, when INT_POLAR = 1, *0* represents interrupt is valid and *1* represents interrupt is invalid.				

The registers irrelevant to FIFO are not discussed in this document.

1.2 FIFO Operating Mode

The FIFO of CMT2219B can be configured as 32-byte or 64-byte.

Usually, it is recommended to preset FIFO operating mode in STBY state before starting Rx. All registers in configuration area and control area 2 can be saved in SLEEP state. Therefore, configuring once is enough unless operating mode change is needed.

- **Preset FIFO**

1. Set DATA_MODE <1:0> as 2, namely, set data process mode as packet mode.

2. Configure FIFO_MERGE_EN and select a desired FIFO size.

3. Set FIFO_AUTO_CLR_DIS as 0 if users need clearing FIFO to 0 automatically when entering Rx each time. Otherwise, set it as 1 and clear FIFO to 0 manually.

Refer to Appendix 1 for code example for FIFO read operation.

1.3 Interrupt Timing for FIFO

The interrupt timing when FIFO is in operating is shown in the below figure.

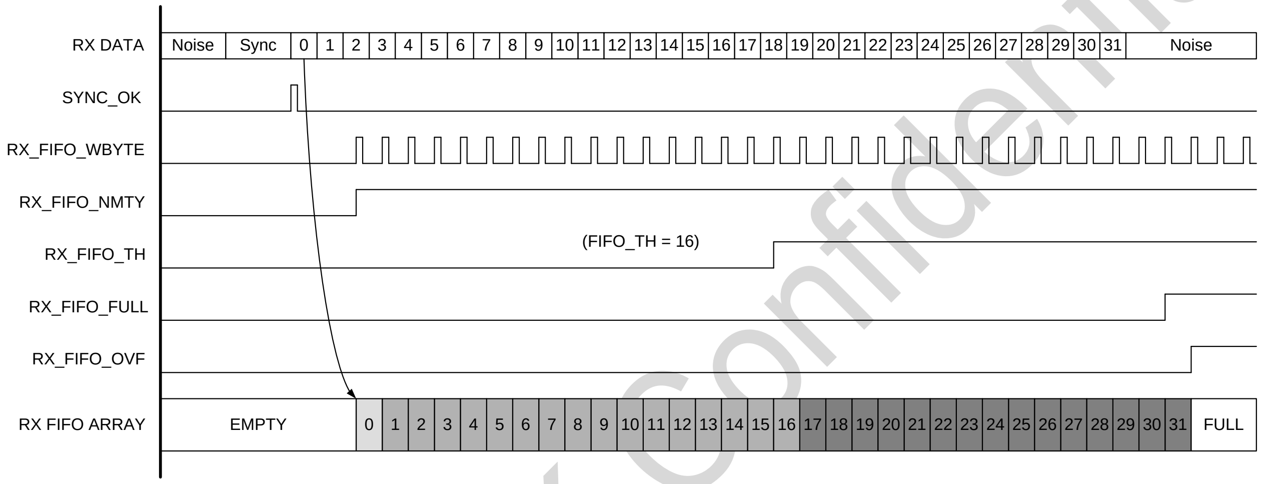


Figure 2. CMT2219B RX FIFO Interrupt Timing Diagram

1.4 FIFO Usage

FIFO starts work after the FIFO configuration is ready. Several typical application scenarios are provided below. Some configurations and controls other than the FIFO configuration are required to accomplish the overall receiving flow, which are discussed in related AN document. This document covers FIFO related content only.

FIFO has a relatively simple operating flow, that is, clear to 0 before entering Rx each time, then input FIFO once data receiving happens (when Sync Word is detected successfully). After the receiving, no extra steps are needed and users just need to clear FIFO by sending FIFO_CLR_RX before the next receiving. Co-working with interrupts, the MCU can fulfill the following operations.

1. Once RX_FIFO_FULL interrupt is detected, which indicates that the FIFO has been fully filled, users can start reading FIFO. It suits for the scenario where the packet length is equal to the FIFO depth and users read the FIFO only after receiving a complete packet.

2. Once RX_FIFO_TH interrupt is detected, which indicates that the filled data in FIFO meets the preset length, users can start reading. It suits for the scenario where the packet length is not equal to the FIFO depth and users read the FIFO only after receiving a complete packet.

3. Once RX_FIFO_NMTY interrupt is detected, users start and keep reading FIFO until the interrupt is not valid. Users wait for interrupt being valid again to have next FIFO reading. This way it can fulfill reading while receiving. It suits for the scenario

where the packet length is larger/shorter than or equal to the FIFO depth.

4. Once RX_FIFO_WBYTE interrupt is detected, users start reading FIFO. This way it can fulfill receiving then reading byte by byte, which should meet a precondition that the SPI speed is faster than the data receiving speed..

If continuous receiving is required, it's suggested to choose the 4th operation mentioned above. It requires the SPI speed is fast enough, saying it should be 1.5 times faster than the speed by which SPI writes one byte into FIFO. For example, if the data rate is 10 kHz, the time to receive a byte is about 800 us. It requires the speed by which SPI reads a byte is 1.5 times faster, that is, the SPI speed is up to 534 us. It takes 8 SCL clock cycles for SPI to read a byte, plus the time overhead before and after, that is, the time to read a byte is 10 SCL clock cycles. That is to say each cycle takes 53.4 us. After conversion, the SCL clock frequency is about 18.7 kHz, roughly twice the data rate.

2 Packet Format

CMT2219B applies a typical and flexible packet format with the packet structure shown in the below figure.

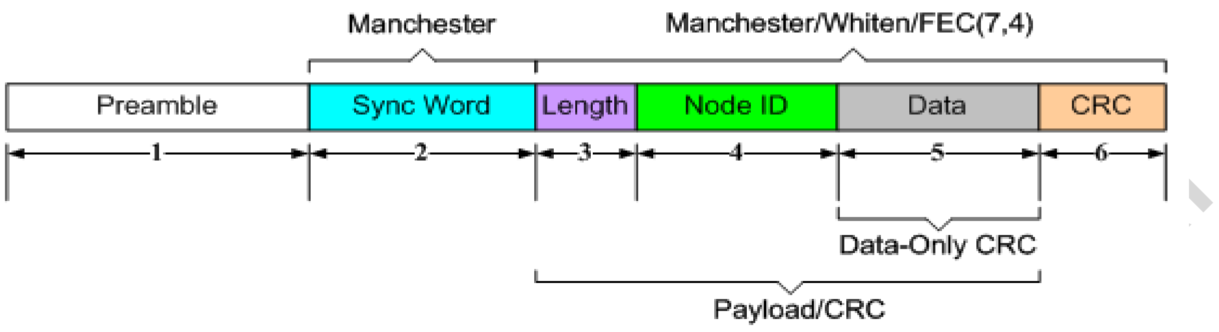


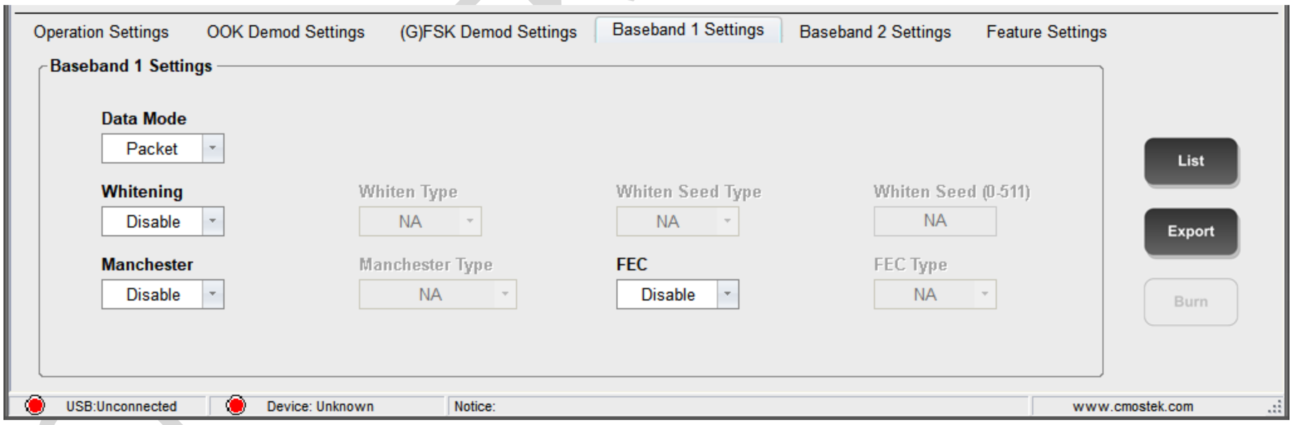
Figure 3. Packet Structure Diagram

The package format contains 6 optional sections, which can meet most single field structure based requirements in the market (it's not compatible with multiple field structure). The configurable content in the package format is mainly located in the baseband *part*. The followings will discuss how to configure each part in RFPDK screen and explain the associated registers.

2.1 Data Mode Configuration

Data Mode refers to the mode based on which the external MCU forms the transmitted data or acquires the received data.

The corresponding RFPDK screen and parameters are shown in the below figure.



Baseband 1 Settings

Data Mode: Packet

Whitening: Disable

Whiten Type: NA

Whiten Seed Type: NA

Whiten Seed (0-511): NA

Manchester: Disable

Manchester Type: NA

FEC: Disable

FEC Type: NA

Figure 4. Data Format Screen in RFPDK

Table 6. Parameters Related to Data Mode

RFPDK Parameter	Register Bit
Data Mode	DATA_MODE <1:0>

Table 7. Registers in Configuration Area

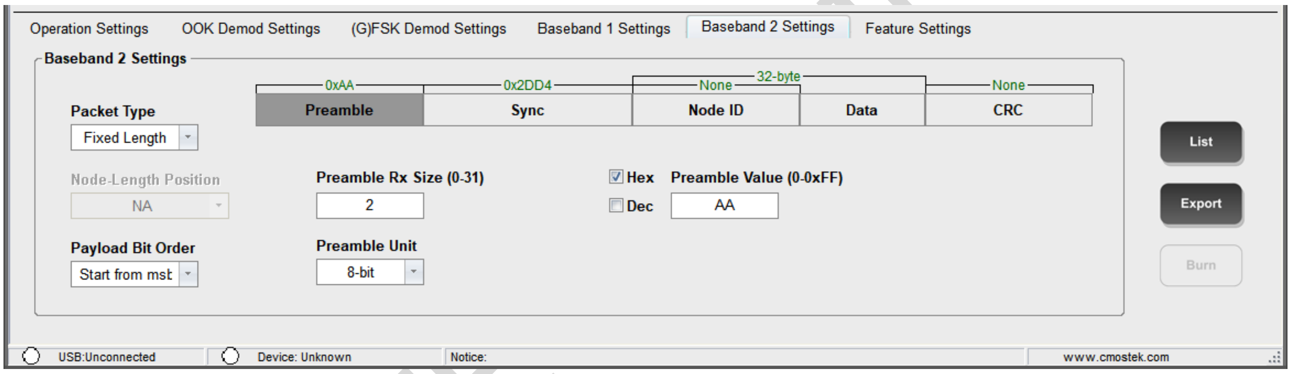
Register	Bit	R/W	Flag	Description
CUS_PKT1 (0x38)	1:0	RW	DATA_MODE<1:0>	Data receiving and transmitting modes selection. 0: direct mode (default) 1: NA 2: packet mode 3: NA

The difference between the 2 data modes is as follows

- Direct - direct mode, supporting preamble and sync detection only, no need for FIFO operating.
- Packet - packet mode, supporting configuring all packet types with FIFO operating required.

2.2 Preamble Configuration

The corresponding RFPDK screen and parameters are shown in the below figure.



Baseband 2 Settings

0xAA	0x2DD4	None	32-byte	None
Preamble	Sync	Node ID	Data	CRC

Packet Type: Fixed Length
Node-Length Position: NA
Payload Bit Order: Start from msb
Preamble Rx Size (0-31): 2
Preamble Unit: 8-bit
[✓ Hex] Dec
Preamble Value (0-0xFF): AA

USB:Unconnected | Device: Unknown | Notice: | www.cmostek.com

Figure 5. Preamble Configuration Screen in RFPDK

Table 8. Preamble Related Parameters

RFPDK Parameter	Register Bit
Preamble Rx Size	RX_PREAM_SIZE<4:0>
Preamble Unit	PREAM_LENG_UNIT
Preamble Value	PREAM_VALUE<7:0>

Table 9. Registers in Configuration Area

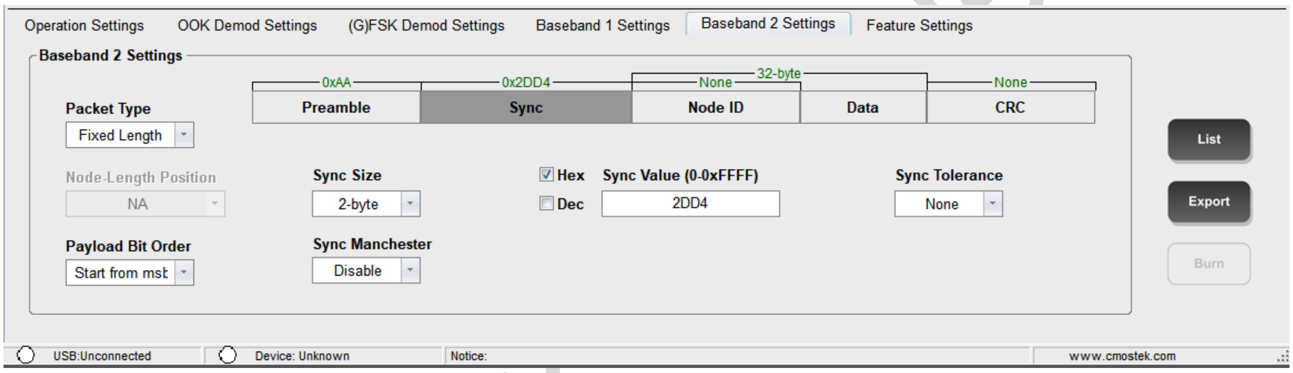
Register	Bit	R/W	Flag	Description
CUS_PKT1 (0x38)	7:3	RW	RX_PREAM_SIZE<4:0>	The preamble length in Rx mode. It can be configured as 0 - 31 units with 0 representing no preamble detection, 1 representing 1-unit preamble being detected, and so on.
	2	RW	PREAM_LENG_UNIT	Preamble length unit.

Register	Bit	R/W	Flag	Description
				0: 8-bit unit 1: 4-bit unit
CUS_PKT4 (0x3B)	7:0	RW	PREAM_VALUE<7:0>	The value of preamble. When PREAM_LEN_UNIT = 0, all the 8 bits are valid. When PREAM_LEN_UNIT = 1, only bits <3:0> are valid.

A successful preamble detection will result in a PREAM_OK interrupt. In addition, preamble detection is performed throughout the data receiving duration. It is recommended that users detect the interrupt only when needed.

2.3 Sync Word Configuration

The corresponding RFPDK parameter and screen are shown in the below figure.



Baseband 2 Settings

Packet Type: Fixed Length

Node-Length Position: NA

Payload Bit Order: Start from msb

Sync Size: 2-byte

Sync Manchester: Disable

☑ Hex ☐ Dec

Sync Value (0-0xFFFF): 2DD4

Sync Tolerance: None

Figure 6. Sync Word Configuration Screen in RFPDK

Table 10. Sync Word Related Registers

RFPDK Parameter	Register Bits
Sync Size	SYNC_SIZE<2:0>
Sync Value	SYNC_VALUE<63:0>
Sync Tolerance	SYNC_TOL<2:0>
Sync Manchester	SYNC_MAN_EN

Table 11. Registers in Configuration Area

Register	Bit	R/W	Flag	Description
CUS_PKT5 (0x3C)	6:4	RW	SYNC_TOL<2:0>	The number of fault-tolerant bits for Sync Word detection in RX mode. 0: no error allowed 1: allow 1 error bit reception. 2: allow 2 error bits reception. 3: allow 3 error bits reception. 4: allow 4 error bits reception.

Register	Bit	R/W	Flag	Description
				5: allow 5 error bits reception. 6: allow 6 error bits reception. 7: allow 7 error bits reception.
	3:1	RW	SYNC_SIZE<2:0>	Sync Word length. 0: 1 byte 1: 2 bytes 2: 3 bytes 3: 4 bytes 4: 5 bytes 5: 6 bytes 6: 7 bytes 7: 8 bytes
	1:0	RW	SYNC_MAN_EN	Manchester encoding and decoding enabling for Sync Word. 0: disabled 1: enable
CUS_PKT6 (0x3D)	7:0	RW	SYNC_VALUE<7:0>	The value of Sync Word. It is filled in different registers according to different SYNC_SIZE settings, see the table below for details.
CUS_PKT7 (0x3E)	7:0	RW	SYNC_VALUE<15:8>	
CUS_PKT8 (0x3F)	7:0	RW	SYNC_VALUE<23:16>	
CUS_PKT9 (0x40)	7:0	RW	SYNC_VALUE<31:24>	
CUS_PKT10 (0x41)	7:0	RW	SYNC_VALUE<39:32>	
CUS_PKT11 (0x42)	7:0	RW	SYNC_VALUE<47:40>	
CUS_PKT12 (0x43)	7:0	RW	SYNC_VALUE<55:48>	
CUS_PKT13 (0x44)	7:0	RW	SYNC_VALUE<63:56>	

Table 12. Value of Sync Word

	SYNC_VALUE							
SYNC_SIZE	<63:56>	<55:48>	<47:40>	<39:32>	<31:24>	<23:16>	<15:8>	<7:0>
0	√							
1	√	√						
2	√	√	√					
3	√	√	√	√				

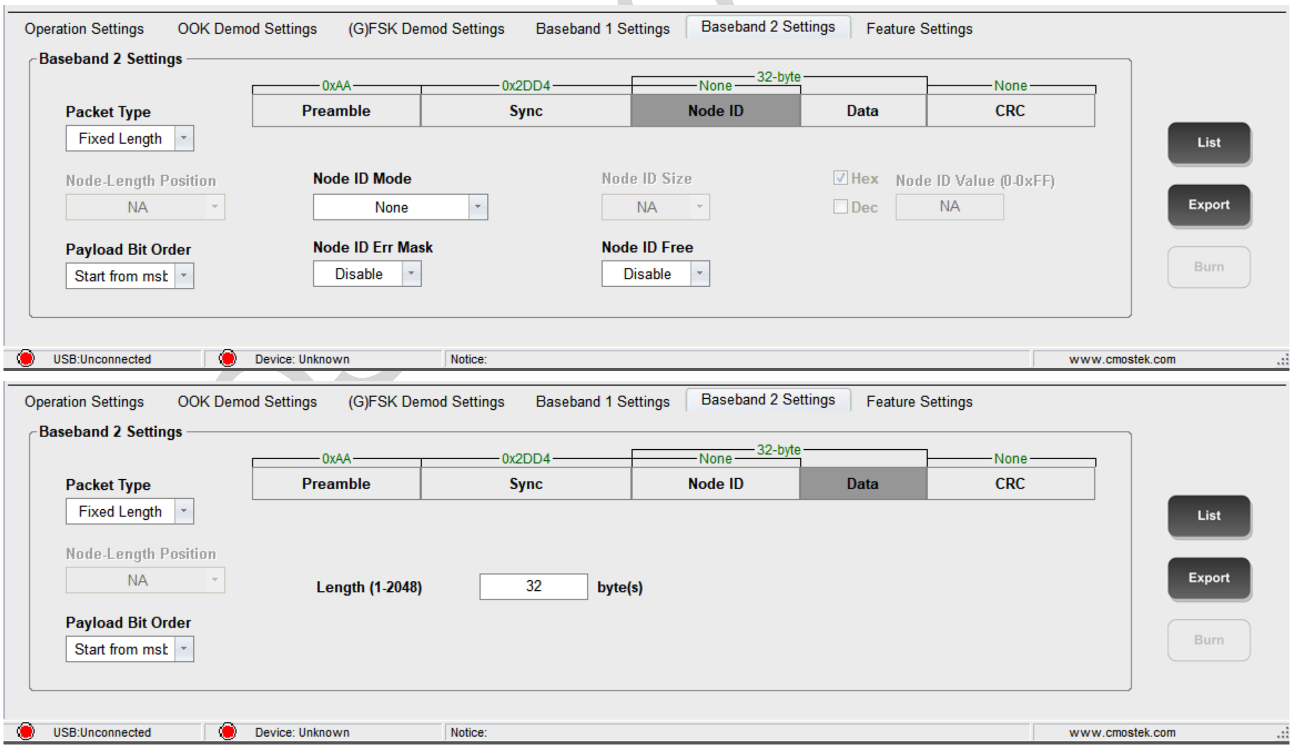
	SYNC_VALUE							
4	√	√	√	√	√			
5	√	√	√	√	√	√		
6	√	√	√	√	√	√	√	
7	√	√	√	√	√	√	√	√

In the above table, the registers ticked need to be filled in. For example, if SYNC_SIZE is set to 1, namely the length is 2 bytes, and the value of the sync word is 0x5678, then users need to fill this value into the two registers, SYNC_VALUE<63:56> and SYNC_VALUE<55:48> with msb and lsb filled into the addresses start with the 63rd bit and the 48th bit respectively, that is, 0x56 is filled in SYNC_VALUE<63:56> and 0x78 in SYNC_VALUE<55:48>.

In addition, some applications require Manchester encoding for entire packet, however most of applications only need payload encoding, therefore a separate Manchester encoding enabling bit is provided specific for Sync Word part.

2.4 Overall Data Packet Configuration

The corresponding RFPDK parameter and screen are shown in the below figure.



Packet Type: Fixed Length
Node-Length Position: NA
Payload Bit Order: Start from msb
Node ID Mode: None
Node ID Err Mask: Disable
Node ID Size: NA
Node ID Free: Disable
Hex ✓ / Dec
Node ID Value (0-0xFF): NA

Packet Type: Fixed Length
Node-Length Position: NA
Payload Bit Order: Start from msb
Length (1-2048): 32 byte(s)

Figure 7. Data Packet Configuration Screen in RFPDK

Table 13. Data Packet Related Parameters

RFPDK Parameter	Register Bit
Packet type	PKT_TYPE
Payload length, which is calculated based on various parameters. See below for more details.	PAYLOAD_LENG<10:0>
Node-length position	NODE_LENG_POS_SEL
Payload bit order	PAYLOAD_BIT_ORDER
No specific parameter in RFPDK for this register. It is used flexibly in real applications. See below for more details.	AUTO_ACK_EN

Table 14. .Registers in Configuration Area

Register	Bit	R/W	Flag	Description
CUS_PKT14 (0x45)	6:4	RW	PAYLOAD_LENG<10:8>	<10:8> bits of the 11-bit payload length. When the PKT_TYPE is set as fixed length packet, the configuration range is 0 - 2047, that is 1 - 2048 bytes. When the PKT_TYPE is set as packet with variable length, only <7:0> is valid, with a configuration range of 1 - 256 bytes.
	2	RW	NODE_LENG_POS_SEL	In packets with variable length, the positions of node ID and length byte 0: node ID is before length byte 1: node ID is after length byte
	1	RW	PAYLOAD_BIT_ORDER	0: for each byte of payload + CRC, encode and decode MSB first 1: for each byte of payload + CRC, encode and decode LSB first.
	0	RW	PKT_TYPE	Packet length type. 0: fixed packet length 1: variable packet length
CUS_PKT15 (0x46)	7:0	RW	PAYLOAD_LENG<7:0>	The <7:0> bits of the 12-bit payload length.

The detail description of PAYLOAD_BIT_ORDER is as below.

PAYLOAD_BIT_ORDER = 1 means that, for each byte of payload and CRC, LSB is sent or encoded with Manchester/Whiten encoding first then the MSB. On the other hand, during receiving, for each byte of the decoded payload, the order of MSB and LSB must be changed, and CRC decoding is performed after then. If the RX and TX have the same configuration, users will not aware of it. When using this product interfacing with products from other providers, users need to understand the process and make the appropriate configuration.

PAYLOAD_BIT_ORDER = 0 means no above processing performed.

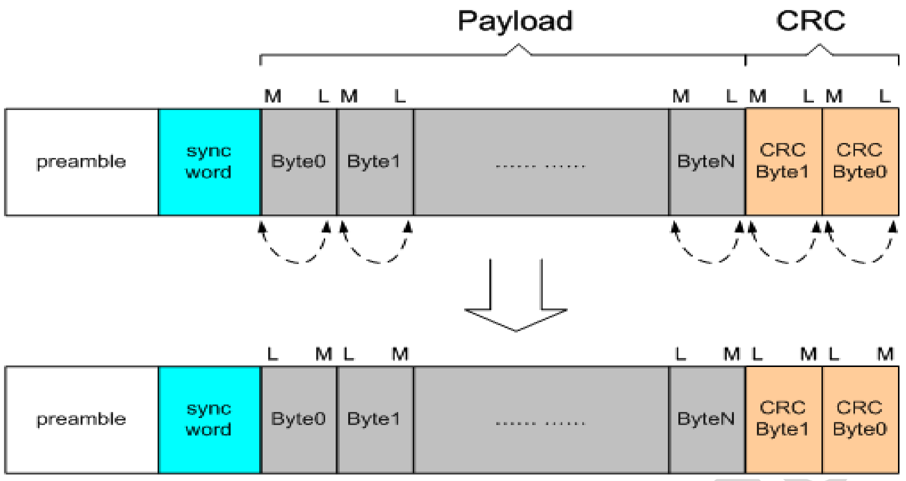
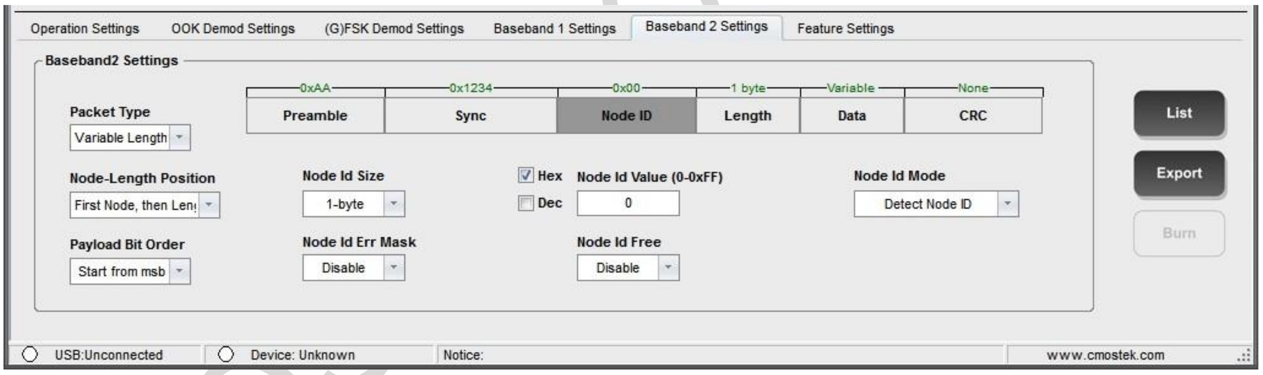


Figure 1. PAYLOAD_BIT_ORDER Operation

2.5 Node ID Configuration

The corresponding RFPDK parameter and screen are shown in the below figure.



Packet Type: Variable Length

Node-Length Position: First Node, then Len

Payload Bit Order: Start from msb

Node Id Size: 1-byte

Node Id Err Mask: Disable

[✓ Hex] Dec

Node Id Value (0-0xFF): 0

Node Id Free: Disable

Node Id Mode: Detect Node ID

Figure 8. Node ID Configuration Screen in RFPDK

Table 15. Node ID Related Parameter

RFPDK Parameter	Register Bit
Node Id Size	NODE_SIZE<1:0>
Node Id Mode	NODE_DET_MODE<1:0>
Node Id Value	NODE_VALUE<31:0>
Node Id Err Mask	NODE_ERR_MASK
Node Id Free	NODE_FREE_EN

Table 16. Registers in Configuration Area

Register	Bit	R/W	Flag	Description
CUS_PKT16 (0x47)	5	RW	NODE_FREE_EN	In RX mode, make the Node ID detection an independent circuit. 0: enable 1: disable
	3	RW	NODE_ERR_MASK	If node ID detection error occurs, an PKT_ERR interrupt will be output, meanwhile the decoding circuit will be reset. This bit is to control whether or not the reset is performed. 0: reset is enabled 1: reset is disabled
	2	RW	NODE_SIZE<1:0>	Node ID length. 0: 1 byte 1: 2 bytes 2: 3 bytes 3: 4 bytes
	1	RW	NODE_DET_MODE<1:0>	Node ID detection mode. 0: no detection 1: TX mode sends the content of NODE_VALUE and RX mode recognizes the content of NODE_VALUE only. 2: TX mode sends the content of NODE_VALUE. RX mode can recognize the content of NODE_VALUE and all 0 only. 3: TX mode sends the content of NODE_VALUE. RX mode can recognize the content of NODE_VALUE, all *0* and all *1*.
CUS_PKT17 (0x48)	7:0	RW	NODE_VALUE<7:0>	The 32-bit Node ID.
CUS_PKT18 (0x49)	7:0	RW	NODE_VALUE<15:8>	
CUS_PKT19 (0x4A)	7:0	RW	NODE_VALUE<23:16>	
CUS_PKT20 (0x4B)	7:0	RW	NODE_VALUE<31:24>	

Table 17. 32-bit Node ID

	NODE_VALUE			
NODE_SIZE	<31:24>	<23:16>	<15:8>	<7:0>
0	√			
1	√	√		
2	√	√	√	
3	√	√	√	√

In the above table, the registers ticked need to be filled in. For example, if NODE_SIZE is set to 1, namely the length is 2 bytes, and the value is 0x5678, then users need to fill this value into the two registers, SYNC_VALUE<31:24> and SYNC_VALUE<23:16> with MSB and LSB filled in addresses start with the 31st bit the 16th bit respectively, that is, 0x56 is filled in SYNC_VALUE<31:24> and 0x78 in SYNC_VALUE<23:16>.

The followings will discuss the PAYLOAD_LENG and NODE_SIZE settings, data payload length determination and the understanding of length byte according to different PKT_TYPE, see below tables for more details

.

- **Fixed packet format**

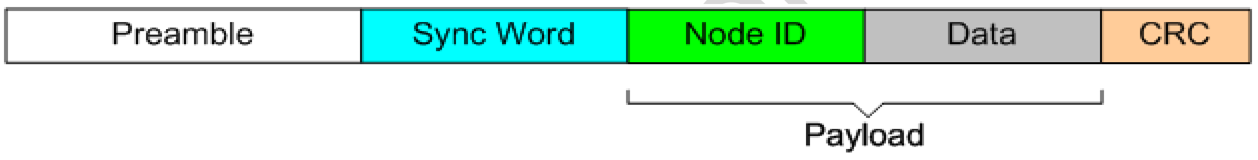


Figure 9. Packet Format for Packets with Fixed Length

Parameter Configuration
Payload structure: Node ID + Data. Node ID can be absent. Same configuration for Tx and Rx. For example, PAYLOAD_LENG<10:0> = 11 NODE_SIZE<1:0> = 2 Then the total length of payload is 11 + 1 = 12, and the length of the Node ID is 2 + 1 = 3, so the length of data is 12 – 3 = 9 bytes. If the Node ID does not exist, the length of data is 12.

- **Format of packets with variable length**

Condition 1：Node ID does not exist

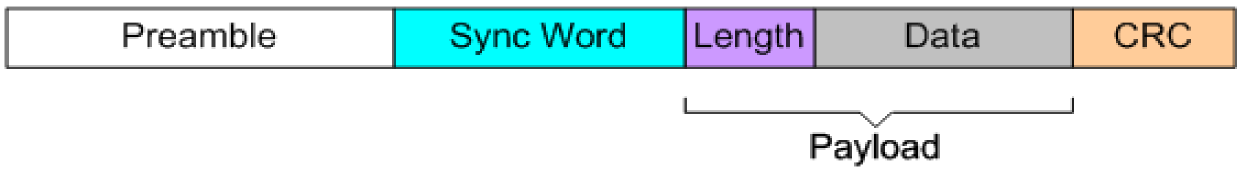


Figure 10. Format of Packets with Variable Length (No Node ID Contained)

Parameter Configuration
Payload structure Length Byte + Data Length byte represents the length of the following data.

Condition 2：Node ID is contained and NODE_POSITION = 0 (Node ID is located before the length)

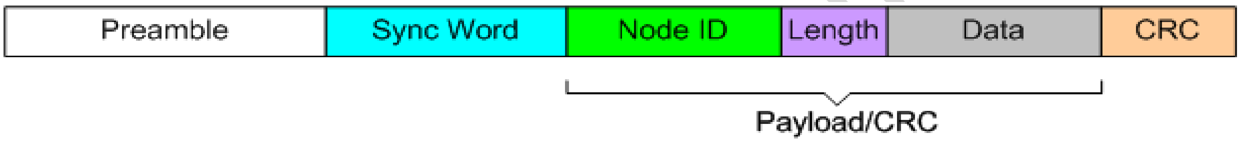


Figure 11. Format of Packets with Variable Length with Node ID Contained (Node ID is located before Length Byte)

RX Mode
Payload structure Node ID + length byte + data Length byte represents the length of the following data.

Condition 3：Node ID is contained and NODE_POSITION = 1 (Node ID is after length byte)

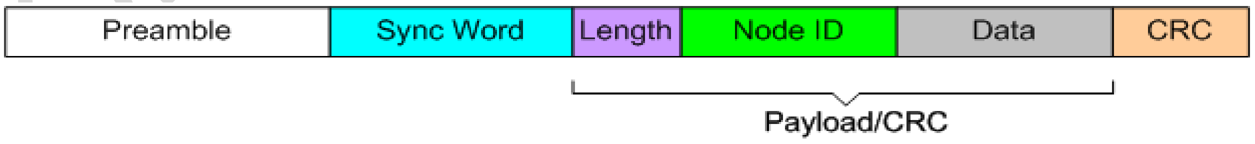
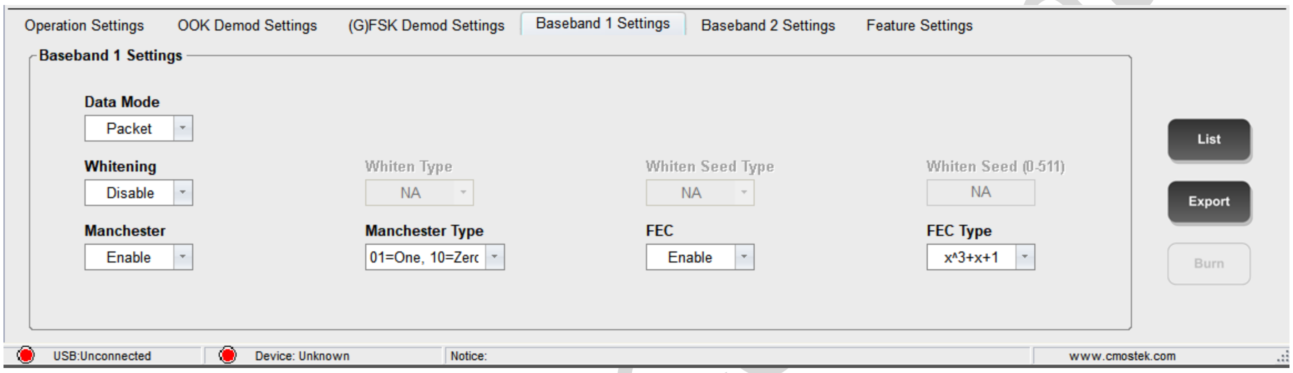


Figure 12. Format of Packets with Variable Length with Node ID Contained (Node ID is located after Length Byte)

Parameter Configuration
Payload structure: Length Byte + Node ID + Data Length byte represents the length of Node ID + Data, corresponding to Tx configuration.

2.6 FEC Configuration

The corresponding RFPDK parameter and screen are shown in the below figure.


Operation Settings | OOK Demod Settings | (G)FSK Demod Settings | Baseband 1 Settings | Baseband 2 Settings | Feature Settings

Baseband 1 Settings

Data Mode: Packet

Whitening: Disable — Whiten Type: NA — Whiten Seed Type: NA — Whiten Seed (0-511): NA

Manchester: Enable — Manchester Type: 01=One, 10=Zerc — FEC: Enable — FEC Type: x^3+x+1

List | Export | Burn

USB:Unconnected | Device: Unknown | Notice: | www.cmostek.com

Figure 13. FEC Configuration Screen in RFPDK

Table 18. FEC Related Configuration

RFPDK Parameter	Register Bit
FEC	FEC_EN
FEC_Type	FEC_TYPE

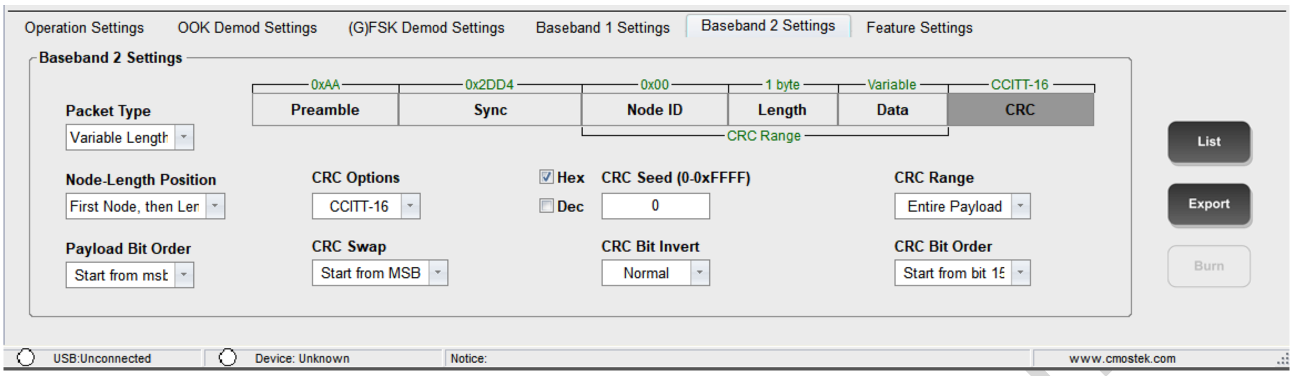
Table 19. Registers in Configuration Area

Register	Bit	R/W	Flag	Description
CUS_PKT21 (0x4C)	7	RW	FEC_TYPE	Polynomial selection for FEC (7, 4) encoding and decoding. 0: the polynomial is x^3+x+1 1: the polynomial is x^3+x^2+1
	6	RW	FEC_EN	FEC(7,4) encoding and decoding enabling. 0: disable 1: enable

The main function of FEC is to correct wrong data in a package resulting in error rate reduction.

2.7 CRC Configuration

The corresponding RFPDK parameter and screen are shown in the below figure.



Baseband 2 Settings

Packet Type	Variable Length
Node-Length Position	First Node, then Ler
Payload Bit Order	Start from msb
CRC Options	CCITT-16
CRC Swap	Start from MSB
CRC Seed (0-0xFFFF)	0
☑ Hex ☐ Dec	
CRC Bit Invert	Normal
CRC Range	Entire Payload
CRC Bit Order	Start from bit 15

Figure 14.. CRC Configuration Screen in RFPDK

Table 20. CRC Related Parameters

RFPDK Parameter	Register Bit
Crc Options	CRC_TYPE<1:0>
Crc Seed	CRC_SEED<15:0>
If crc options is None, CRC_EN = 0. Otherwise, CRC_EN =1	CRC_EN
Crc Range	CRC_RANGE
Crc Swap	CRC_BYTE_SWAP
Crc Bit Inv	CRC_BIT_INV
Crc Bit Order	CRC_BIT_ORDER

Table 21. Registers in Configuration Area

Register	Bit	R/W	Flag	Description
CUS_PKT21 (0x4C)	5	RW	CRC_BYTE_SWAP	CRC transmission and reception sequence. 0: send and receive higher byte first 1: send and receive lower byte first
	4	RW	CRC_BIT_INV	Whether the CRC code is reversed. 0: CRC code is not reversed 1: CRC code is reversed bit by bit
	3	RW	CRC_RANGE	CRC calculation range. 0: the entire payload 1: data part only
	2:1	RW	CRC_TYPE<1:0>	CRC polynomial type.

Register	Bit	R/W	Flag	Description
				0: CCITT-16 1: IBM-16 2:: ITU-16 (equivalent to reverse CCITT-16) 3: NA
	0	RW	CRC_EN	CRC enabling. 0: disable 1: enable
CUS_PKT22 (0x4D)	7:0	RW	CRC_SEED<7:0>	The initial value of CRC polynomial.
CUS_PKT23 (0x4E)	7:0	RW	CRC_SEED<15:8>	
CUS_PKT24 (0x4F)	7	RW	CRC_BIT_ORDER	Big or little endian selection of CRC 0: CRC bytes are sent and received in the order from bit 15 to bit 0. 1: CRC bytes are sent and received in the order from bit 0 to bit 15.

Followings are descriptions for some of the CRC configuration parameters.

- **CRC_RANGE**

This parameter is to specify the checked object of CRC, which can be the entire payload or just the data part.

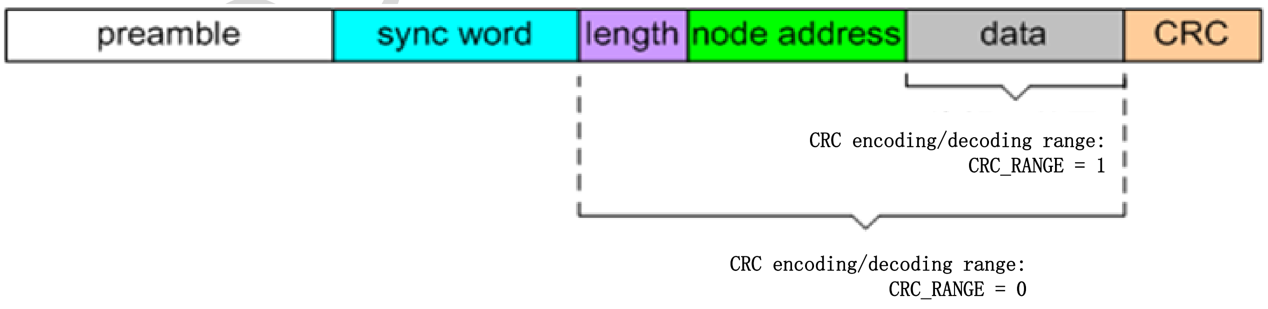


Figure 15. CRC Encoding Range

- **CRC_BIT_INV**

This parameter is to control the function of logical negation, that is, making 0 become 1 and 1 become 0.

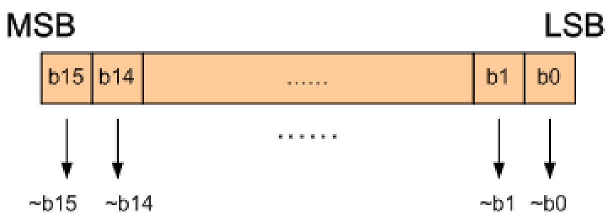


Figure 16.2 CRC_BIT_INV

- **CRC_BYTE_SWAP**

This function is to reverse the position of two bytes, but does not change the bit order in each byte.

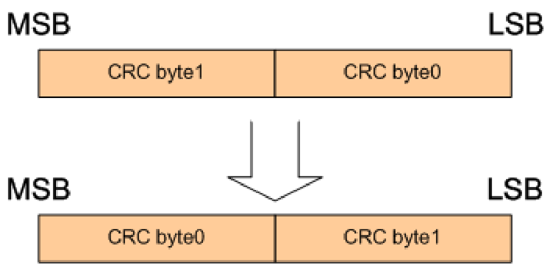


Figure 17. CRC_BYTE_SWAP

- **CRC_BIT_ORDER**

This function is to reverse the entire bit order of CRC part. If the positions of the two bytes are swapped, it reverse the bit order based on the bytes after swap.

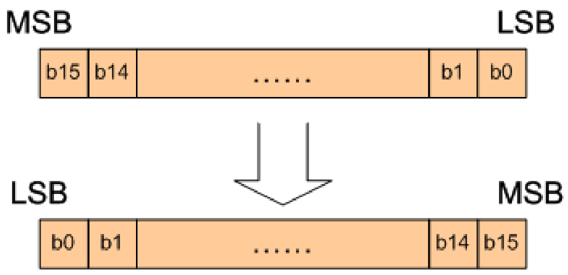
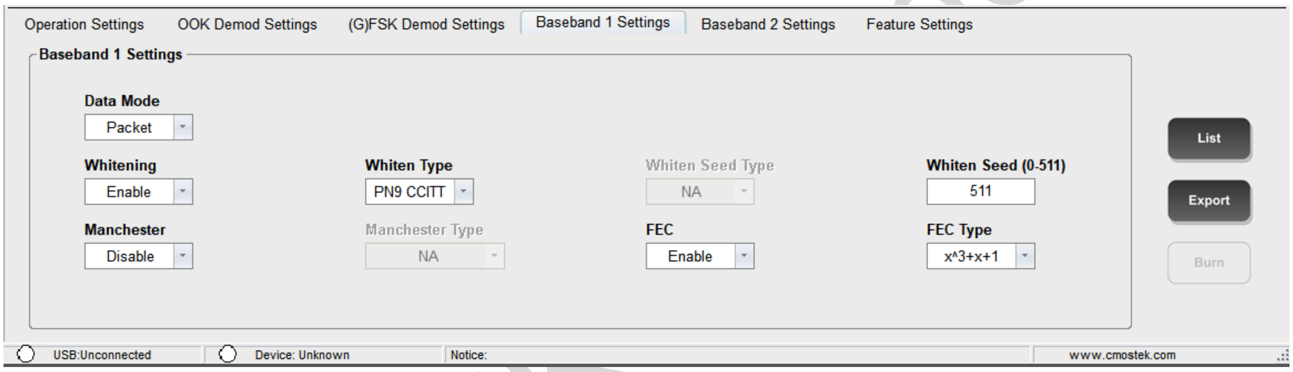


Figure 18. CRC_BIT_ORDER

2.8 Encoding and Decoding Configuration

The corresponding RFPDK parameter and screen are shown in the below figure.



Baseband 1 Settings

Data Mode: Packet

Whitening: Enable

Whiten Type: PN9 CCITT

Whiten Seed Type: NA

Whiten Seed (0-511): 511

Manchester: Disable

Manchester Type: NA

FEC: Enable

FEC Type: x^3+x+1

Figure 19. Encoding and Decoding Configuration Screen in RFPDK

Table 22. Encoding and Decoding Related Parameters

RFPDK Parameter	Register Bit
Whitening	WHITEN_EN
Whiten Type	WHITEN_TYPE<1:0>
Whiten Seed Type	WHITEN_SEED_TYPE
Whiten Seed	WHITEN_SEED<8:0>
Manchester	MANCH_EN
Manchester Type	MANCH_TYPE

Table 23. Registers in Configuration Area

Register	Bit	R/W	Flag	Description

Register	Bit	R/W	Flag	Description
CUS_PKT24 (0x4F)	6	RW	WHITEN_SEED<8>	The 8th bit of the WHITEN encoding and decoding seed polynomial. When the WHITEN encoding and decoding is PN9, the seed uses all the 9 bits. When it is PN7, the seed uses the lower 7 bits.
	5	RW	WHITEN_SEED_TYPE	The seed type when the WHITEN encoding and decoding polynomial is PN7 0: the PN7 seed is calculated based on the A7139 method 1: PN7 seed is the value defined by whiten_seed
	4:3	RW	WHITEN_TYPE<1:0>	Whitening encoding and decoding methods. 0: PN9 CCITT encoding and decoding 1: PN9 IBM encoding and decoding 2: PN7 encoding and decoding 3: Invalid
	2	RW	WHITEN_EN	Whitening encoding and decoding enabling. 0: disable 1: enable
	1	RW	MANCH_TYPE	Manchester encoding and decoding methods. 0: 01 represents 1 and 10 represents 0 1: 10 represents 1 and 01 represents 0
	0	RW	MANCH_EN	Manchester encoding and decoding enabling. 0: disable 1: enable
CUS_PKT25 (0x50)	7:0	RW	WHITEN_SEED<7:0>	The 7:0 bits of the WHITEN encoding and decoding seed polynomial.

3 GPIO and Interrupt

The CMT2219B provides 3 GPIOs. Each of them can be configured as different input or output. It provides 2 interrupt ports ,which can be configured to different GPIO outputs. See the following chapters for more details.

3.1 GPIO Configuration

The GPIO related registers in configuration area 1 are listed in the below table.

Table 24. GPIO Related Registers

Register	Bit	R/W	Flag	Description
CUS_IO_SEL (0x65)	5:4	RW	GPIO3_SEL<1:0>	GPIO3 selection. 0: CLKO 1: DOUT 2: INT2 3: DCLK
	3:2	RW	GPIO2_SEL<1:0>	GPIO2 selection. 0: INT1 1: INT2 2: DOUT 3: DCLK
	1:0	RW	GPIO1_SEL<1:0>	GPIO1 selection. 0: DOUT 1: INT1 2: INT2 3: DCLK

3.2 Interrupt Configuration and Mapping

The CMT2219B provides 2 interrupt ports, INT1 and INT2, which can be assigned to different GPIOs. The interrupt related registers are listed in the below table.

Table 25. Control Area 1

Register	Bit	R/W	Flag	Description
CUS_INT1_CTL (0x66)	5	RW	INT_POLAR	Interrupt output polarity selection. 0: 0 is invalid and 1 is valid 0: 0 is valid and 1 is invalid
	4:0	RW	INT1_SEL<4:0>	For the mapping options of INT1, please refer to the interrupt mapping table below.
CUS_INT2_CTL (0x67)	4:0	RW	INT2_SEL<4:0>	For the mapping options of INT2, please refer to the interrupt mapping table below.
CUS_INT_EN (0x68)	7	RW	SL_TMO_EN	Sleep timeout interrupt enabling. 0: disable 1: enable
	6	RW	RX_TMO_EN	Receiving timeout interrupt enabling. 0: disable 1: enable
	4	RW	PREAM_OK_EN	Successful preamble detection interrupt enabling. 0: disable

Register	Bit	R/W	Flag	Description
				1: enable
	3	RW	SYNC_OK_EN	Successful sync word detection interrupt enabling. 0: disable 1: enable
	2	RW	NODE_OK_EN	Successful Node ID detection interrupt enabling. 0: disable 1: enable
	1	RW	CRC_OK_EN	Successful CRC detection interrupt enabling. 0: disable 1: enable
	0	RW	PKT_DONE_EN	Complete packet reception (right or wrong) interrupt enabling. 0: disable 1: enable
CUS_INT_CLR1 (0x6A)	5	RW	SL_TMO_FLG	SL_TMO interrupt flag
	4	RW	RX_TMO_FLG	RX_TMO interrupt flag
	1	RW	SL_TMO_CLR	Clearing to 0 per SL_TMO interrupt. 0: no action 1: clear to 0
	0	RW	RX_TMO_CLR	Clearing to 0 per RX_TMO interrupt. 0: no action 1: clear to 0

Table 26. Control Area 2

Register	Bit	R/W	Flag	Description
CUS_INT_CLR2 (0x6B)	5	RW	LBD_CLR	LBD active (successful low voltage detection) interrupt clearing. 0: no action 1: clear
	4	RW	PREAM_OK_CLR	PREAM_OK interrupt clearing. 0: no action 1: clear
	3	RW	SYNC_OK_CLR	SYNC_OK interrupt clearing. 0: no action 1: clear
	2	RW	NODE_OK_CLR	NODE_OK interrupt clearing. 0: no action 1: clear
	1	RW	CRC_OK_CLR	CRC_OK interrupt clearing. 0: no action 1: clear
	0	RW	PKT_DONE_CLR	PKT_DONE interrupt clearing.

Register	Bit	R/W	Flag	Description
				0: no action 1: clear
CUS_INT_FLAG (0x6D)	7	RW	LBD_FLG	LBD active (successful low voltage detection) interrupt flag.
	6	RW	COL_ERR_FLG	COL_ERR interrupt flag.
	5	RW	PKT_ERR_FLG	PKT_ERR interrupt flag.
	4	RW	PREAM_OK_FLG	PREAM_OK interrupt flag.
	3	RW	SYNC_OK_FLG	SYNC_OK interrupt flag.
	2	RW	NODE_OK_FLG	NODE_OK interrupt flag.
	1	RW	CRC_OK_FLG	CRC_OK interrupt flag.
	0	RW	PKT_OK_FLG	PKT_OK interrupt flag.

Table 27. Interrupt Mapping Table of CMT2219B

Name	INT1_SEL	Description	Clearing Method
RX_ACTIVE	00000	This interrupt indicates preparing to enter RX or being in RX already. The interrupt value is 1 when it performs PLL correction or in RX state. It is 0 in the rest cases.	Auto
RSSI_VLD	00010	This interrupt indicates whether RSSI is valid	Auto
PREAM_OK	00011	This interrupt indicates successful reception of preamble.	by MCU
SYNC_OK	00100	This interrupt indicates successful reception of Sync Word.	by MCU
NODE_OK	00101	This interrupt indicates successful reception of Node ID.	by MCU
CRC_OK	00110	This interrupt indicates successful CRC check.	by MCU
PKT_OK	00111	This interrupt indicates successful reception of a complete data packet.	by MCU
SL_TMO	01000	This interrupt indicates SLEEP counter time out.	by MCU
RX_TMO	01001	This interrupt indicates Rx counter time out.	by MCU
RX_FIFO_NMTY	01011	This interrupt indicates Rx FIFO is not empty.	Auto
RX_FIFO_TH	01100	This interrupt indicates the length of unread Rx FIFO content exceeds FIFO TH.	Auto
RX_FIFO_FULL	01101	This interrupt indicates Rx FIFO is full.	Auto
RX_FIFO_WBYTE	01110	This interrupt indicates a byte is filled in Rx FIFO, which is a pulse.	Auto
RX_FIFO_OVF	01111	This interrupt indicates Rx FIFO is full.	Auto
STATE_IS_STBY	10011	This interrupt indicates it is currently in STBY state.	Auto
STATE_IS_FS	10100	This interrupt indicates it is currently in RFS or TFS state.	Auto
STATE_IS_RX	10101	This interrupt indicates it is currently in Rx state.	Auto
LBD	10111	This interrupt indicates LBD is active (VDD is lower than set TH) .	Auto
PKT_DONE	11001	This interrupt indicates complete reception of current data packet. There are four cases. 1. Receive a data packet completely 2.Manchester decoding error occurs and decoding circuit restarts automatically. 3. NODE ID reception error occurs and decoding circuit restarts	by MCU

Name	INT1_SEL	Description	Clearing Method
		automatically. 4. Signal conflict is detected. Wait for further processing from MCU without decoding circuit restart.	

For Interrupt, 1 is active by default. However, it can make 0 active by setting the INT_POLAR register bit to 1. Using INT1 as an example, the below figure shows the interrupt source control and selection. The control and mapping of INT1 and INT2 are the same.

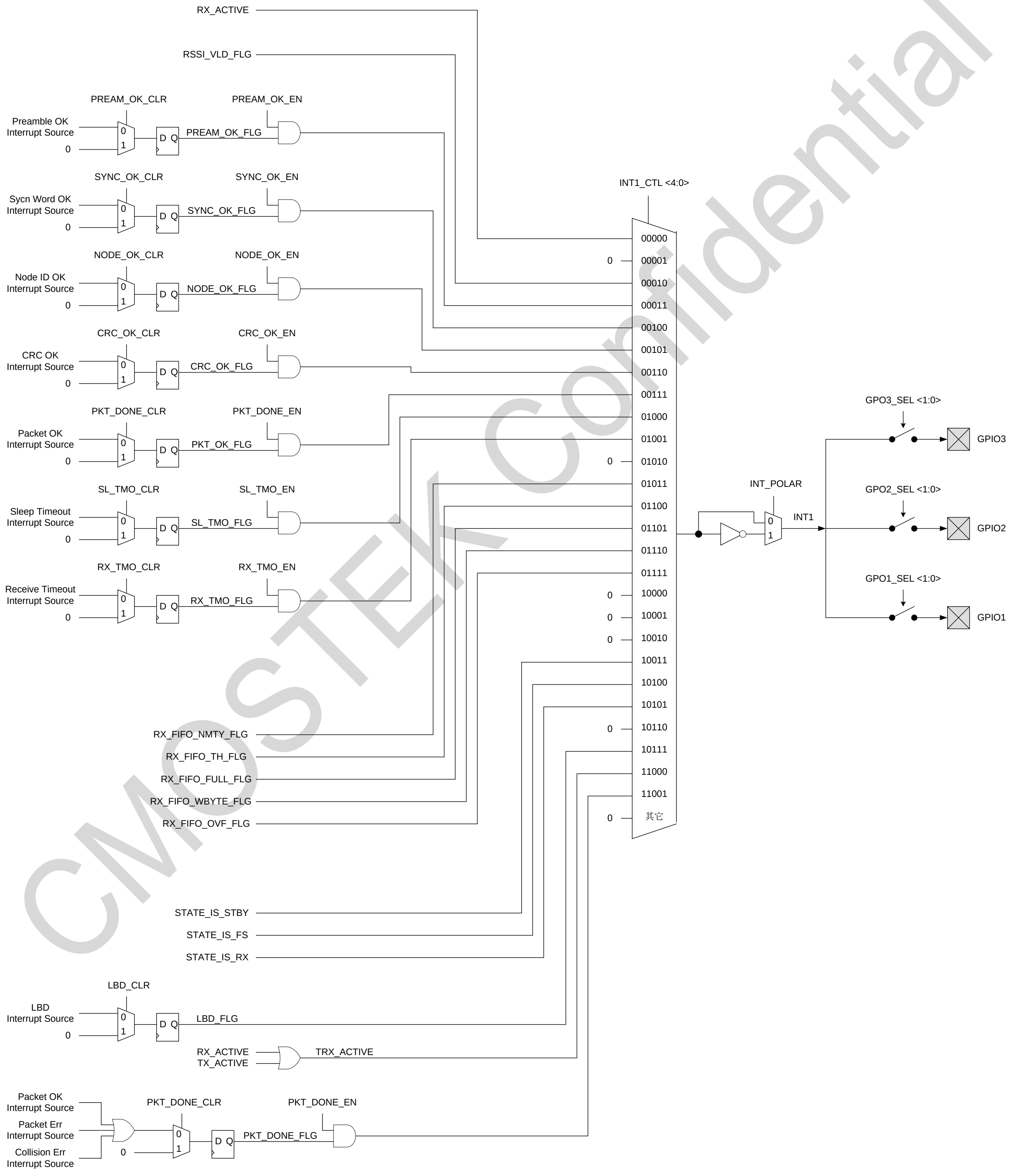


Figure 20. CMT2219B INT1 Interrupt Mapping Diagram

Both an enabling bit (EN) and a clearing bit (CLR) are provided for interrupt sources requiring MCU clearing except LBD, which does not have EN bit. For example, the SYNC_OK interrupt source is generated only when the SYNC_OK_EN bit is set to 1. When this interrupt is generated, it can be cleared only by setting SYNC_OK_CLR to 1. When setting the CLR bit, the MCU only needs to set it to 1 with no need for setting it back to 0, since the CLR bit will be automatically cleared internally once the corresponding interrupt is cleared.

The difference between the two interrupts PKT_OK and PKT_DONE is that PKT_OK represents the packet has been completely received, which is one of the cases of PKT_DONE. However, in practical applications, when a SYNC WORD id detected successfully, it may encounter the below 3 unexpected cases.

1. If NODE ID is contained in a packet, a NODE ID error may be detected. At this time, the PKT_ERR_FLG flag will be set, the chip will stop receiving packets, and automatically restart the decoder, waiting for the next SYNC WORD. In this case, PKT_OK will not be generated.
2. If Manchester decoding is enabled, a decoding error may occur. At this time, the PKT_ERR_FLG flag will be set, the chip will stop receiving packets and automatically restart the decoder, waiting for the next SYNC WORD. In this case, PKT_OK will not be generated.
3. If signal collision detection is enabled (more details will be discussed below), a signal conflict may be detected, which may result in errors in further received packets. At this time, the COL_ERR_FLG flag will be set, and the chip will continue packet receiving until it completes without automatic decoder restart. In this case, PKT_OK is still generated, however the data received is wrong.
4. No matter which of above cases happens, it needs to inform the external MCU, otherwise the MCU will wait for the interrupts all the time resulting in interaction error between the chip and MCU. Therefore, once any one of the above cases happens, the interrupt signal PKT_DONE = PKT_OK | PKT_ERR_FLG | COL_ERR_FLG should be generated to inform the MCU. When receiving the interrupt, the MCU can query the 3 relevant flag bits to know what happened and then have subsequent processing.

Another processing way is that the MCU waits for PKT_OK only, once it is received, check the CRC (if any). However users should coordinate PKT_OK with SYNC_OK interrupt processing properly. For example, if the first packet has a decoding error, the decoder immediately receives the next packet. The MCU may receive two consecutive SYNC_OK interrupts, if the second one is recognized as a PKT_OK interrupt, it will encounter error.

In short, when using encoding and decoding interrupts, due to the complex situations mentioned that may cause misunderstandings, it is recommended for users to make application program as simple as possible. Better user application provides timeout mechanism to ensure stability. If the MCU fails to interact with the chip (disconnected), it may result in a chip crash.

The CMT2219B GPIO output interrupt configuration code example is shown in Appendix 2.

4 Appendix

Appendix 1. Sample Code for FIFO Read and Write Operation

Below is the sample code for FIFO read enabling sub-function.

```
/*! ********************************************************
* @name    Cmt2219B_EnableReadFifo
* @desc    Enable SPI to read the FIFO.
* *********************************************************/
void Cmt2219B_EnableReadFifo(void)
{
   u8 tmp = Cmt2219B_ReadReg(CMT2219B_CUS_FIFO_CTL);
   tmp &= ~CMT2219B_MASK_SPI_FIFO_RD_WR_SEL;
   tmp &= ~CMT2219B_MASK_FIFO_RX_TX_SEL;
   Cmt2219B_WriteReg(CMT2219B_CUS_FIFO_CTL, tmp);
}

/*! ********************************************************
* @name    Cmt2219B_EnableWriteFifo
* @desc    Enable SPI to write the FIFO.
* *********************************************************/
void Cmt2219B_EnableWriteFifo(void)
{
   u8 tmp = Cmt2219B_ReadReg(CMT2219B_CUS_FIFO_CTL);
   tmp |= CMT2219B_MASK_SPI_FIFO_RD_WR_SEL;
   tmp |= CMT2219B_MASK_FIFO_RX_TX_SEL;
   Cmt2219B_WriteReg(CMT2219B_CUS_FIFO_CTL, tmp);
}
```

Appendix 2. Sample Code for GPIO Output Interrupt Configuration Function

```
void RF_Config(void)
{
#ifdef ENABLE_ANTENNA_SWITCH
   /* If you enable antenna switch, GPIO1/GPIO2 will output RX_ACTIVE/TX_ACTIVE,
      and it can't output INT1/INT2 via GPIO1/GPIO2 */
   Cmt2219B_EnableAntennaSwitch(0);
#else
   /* Config GPIOs */
```

```
    Cmt2219B_ConfigGpio(
        CMT2219B_GPIO1_SEL_INT1 |  /* INT1 > GPIO1 */
        CMT2219B_GPIO2_SEL_INT2 |  /* INT2 > GPIO2 */
        CMT2219B_GPIO3_SEL_DOUT  /*DOUT>GPIO3*/
        );

    /* Config interrupt */
    Cmt2219B_ConfigInterrupt(
        CMT2219B_INT_SEL_TX_DONE, /* Config INT1 */
        CMT2219B_INT_SEL_PKT_OK   /* Config INT2 */
        );
#endif

    /* Enable interrupt */
    Cmt2219B_EnableInterrupt(
        CMT2219B_MASK_TX_DONE_EN  |
        CMT2219B_MASK_PREAM_OK_EN |
        CMT2219B_MASK_SYNC_OK_EN  |
        CMT2219B_MASK_NODE_OK_EN  |
        CMT2219B_MASK_CRC_OK_EN   |
        CMT2219B_MASK_PKT_DONE_EN
        );
    Cmt2219B_EnableLfosc(FALSE);
    /* Use a single 64-byte FIFO for either Tx or Rx */
    //Cmt2219B_EnableFifoMerge(TRUE);
    //Cmt2219B_SetFifoThreshold(16);

    /* Go to sleep for configuration to take effect */
    Cmt2219B_GoSleep();
}
```

5 Revise History

Table 28. Revise History Records

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

6 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