CMT2310A自动收发功能使用指南

概要

本文介绍 CMT2310A 自动控制功能的配置及使用方法。

本文档涵盖的产品型号如下表所示。

表 1. 本文档涵盖的产品型号

产品型号	工作频率	调制方式	主要功能	配置方式	封装
CMT2310A	119 - 1050MHz	(4)(G)FSK/OOK	收发一体机	寄存器	QFN24

阅读此文档之前，建议阅读《AN237 CMT2310A 快速上手指南》以了解 CMT2310A 的基本使用方式。

目 录

1. Duty-Cycle 运转模式

1.1 Duty-Cycle 模式相关的寄存器

对应的 RFPDK 的界面和参数如下。

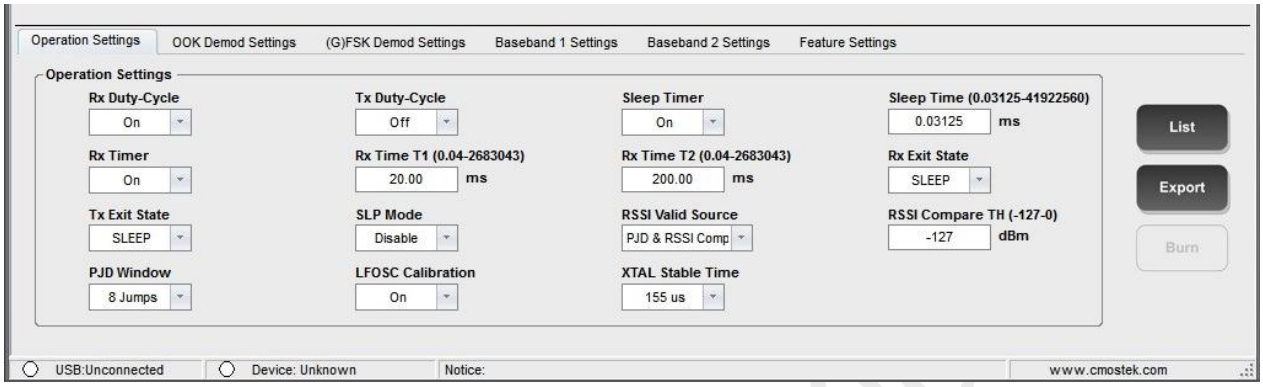


图 1. Duty-Cycle 的 RFPDK 界面

表 2. Duty-Cycle 相关参数

寄存器比特 RFPDK 参数	寄存器比特
Rx Duty-Cycle	RX_DC_EN
Tx Duty-Cycle	TX_DC_EN
Sleep Timer	SLEEP_TIMER_EN
Sleep Time	SLEEP_TIMER_M<10:0> SLEEP_TIMER_R<3:0>
Rx Timer	RX_TIMER_EN
Rx Time T1	RX_TIMER_T1_M<10:0> RX_TIMER_T1_R<3:0>
Rx Time T2	RX_TIMER_T2_M<10:0> RX_TIMER_T2_R<3:0>
Rx Exit State	RX_EXIT_STATE<1:0>
Tx Exit State	TX_EXIT_STATE<1:0>
LFOSC Calibration	LFOSC_RECAL_EN LFOSC_CAL1_EN LFOSC_CAL2_EN
XTAL Stable Time	XTAL_STB_TIME<2:0>

寄存器内容与描述详见下表。

表 3. 位于配置区的寄存器

寄存器名	位数	R/W	比特名	功能说明
CTL_REG_96 (0x60)	6:4	RW	TX_EXIT_STATE<2:0>	完成发射后自动退出到设定的状态： 1：SLEEP 2：READY 3：TFS 4：TX 5：RFS 6：RX Others：SLEEP 只在 Packet 模式下发射完成后才会自动退出 TX，否则芯片会等待 MCU 发 go_* 命令来切换状态。
	0	RW	TX_DC_EN	TX Duty Cycle 的使能： 0：不使能 1：使能
CTL_REG_97 (0x61)	6:4	RW	RX_EXIT_STATE<2:0>	完成接收后自动退出到设定的状态： 1：SLEEP 2：READY 3：TFS 4：TX 5：RFS 6：RX Others：SLEEP 只在 Packet 模式下才会在接收完成后自动退出 RX，否则芯片会等待 MCU 发 go_* 命令来切换。
	3	RW	RX_TIMER_EN	RX TIMER 的使能： 0：不使能 1：使能
	0	RW	RX_DC_EN	RX Duty Cycle 的使能： 0：不使能 1：使能
CTL_REG_99 (0x63)	7:0	RW	SLEEP_TIMER_M<7:0>	定义了 SLEEP TIMER 的计时时间，公式如下： T = M x 2^(R+1) x 31.25 us R 的取值范围是 0-26。
CTL_REG_100 (0x64)	7:5	RW	SLEEP_TIMER_M<10:8>	
	4:0	RW	SLEEP_TIMER_R<4:0>	
CTL_REG_101 (0x65)	7:0	RW	RX_TIMER_T1_M<7:0>	定义了 RX T1 TIMER 的计时时间，公式如下： T = M x 2^(R+1) x 20 us R 的取值范围是 0-21。
CTL_REG_102 (0x66)	7:5	RW	RX_TIMER_T1_M<10:8>	
	4:0	RW	RX_TIMER_T1_R<4:0>	
CTL_REG_103 (0x67)	7:0	RW	RX_TIMER_T2_M<7:0>	定义了 RX T2 TIMER 的计时时间，公式如下： T = M x 2^(R+1) x 20 us
CTL_REG_104	7:5	RW	RX_TIMER_T2_M<10:8>	

寄存器名	位数	R/W	比特名	功能说明
(0x68)	4:0	RW	RX_TIMER_T2_R<4:0>	R 的取值范围是 0-21。
CTL_REG_105 (0x69)	3	RW	SLEEP_TIMER_EN	SLEEP TIMER 的使能： 0：不使能 1：使能

1.2 RX 的 Duty-Cycle 模式

要控制 RX 的 Duty-Cycle 模式，共需要使用 4 个寄存器控制位。下面结合实际应用需求，列出这 4 个控制位的 5 种组合，以及对应的运转模式。

以下状态运行图中，灰色的线代表需要 MCU 手动发送 go_*命令才能切换状态，蓝色的线代表芯片自动完成状态切换。需要注意的是，从状态 A 切换到状态 B，一旦蓝色的线存在，灰色的线就不能存在，反之亦然。即如果采用了自动控制，就不允许 MCU 采用手动控制，否则会打乱芯片的运行，可能会出现死机状况；同样，采用了手动切换，就不能开启自动切换。

1.2.1 全手动控制

全手动控制就是不开启任何 Duty-Cycle 的控制。

表 4. 全手动控制

控制位	值
SLEEP_TIMER_EN	0
RX_DC_EN	0
RX_TIMER_EN	0
RX_EXIT_STATE	0

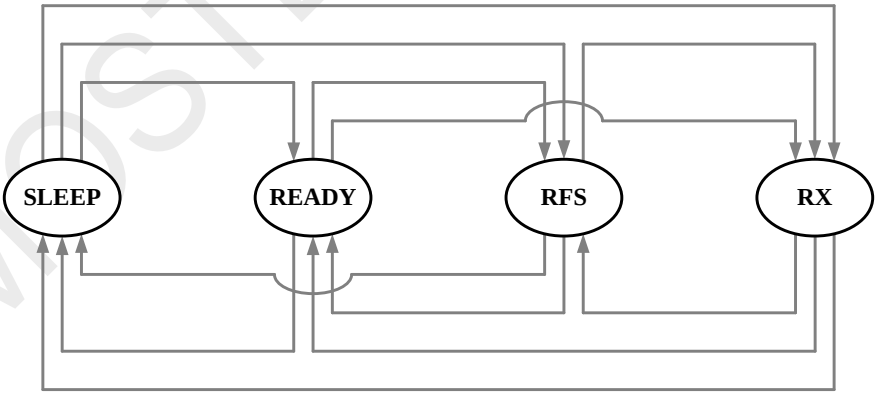


图 2. 接收机全手动控制

1.2.2 自动 SLEEP 唤醒

这种模式只打开 SLEEP TIMER，自动唤醒之后，芯片会切换到 READY，等待 MCU 操作。基于上面

介绍的操作原则，一旦芯片进入了 SLEEP，MCU 就不能发命令去跳出 SLEEP，只能等待睡眠计数器超时，检测到 SL_TMO 中断后才能开始操作。

表 5.自动 SLEEP 唤醒

控制位	值
SLEEP_TIMER_EN	1
RX_DC_EN	0
RX_TIMER_EN	0
RX_EXIT_STATE	0

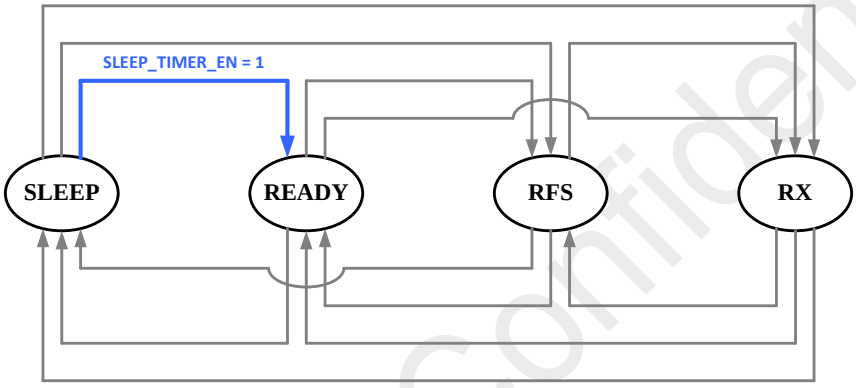


图 3. 接收机自动睡眠唤醒

1.2.3 自动 SLEEP 唤醒，自动进入 RX

将 RX_DC_EN 打开，当芯片自动 SLEEP 唤醒后，就不会跳转到 READY，而是会直接进行 PLL 校正并切换到 RX 进行接收。在这种模式下，RFS 状态不允许使用，MCU 可以参与的操作大量减少。

表 6. 自动 SLEEP 唤醒，自动进入 RX

控制位	值
SLEEP_TIMER_EN	1
RX_DC_EN	1
RX_TIMER_EN	0
RX_EXIT_STATE	0

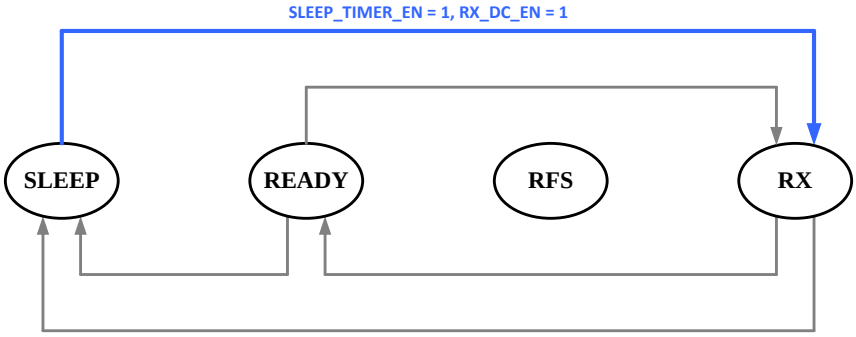


图 4. 接收机自动睡眠唤醒，自动进入接收

1.2.4 自动 SLEEP 唤醒，自动退出 RX

这种模式下，在自动 SLEEP 唤醒的基础上，加上了自动退出 RX，从 READY 进入 RFS 或者 RX 仍然需要 MCU 参与手动切换（即不会自动进入接收），但是一旦进入了 RX，RX TIMER 就会开始计数，超时退出后，会根据不同的 RX_EXIT_STATE 配置自动切换到对应的状态。通常情况下，READY 状态可以被用作一个“中转站”，让 MCU 可以参与操作，例如清除中断和读 FIFO 等。

表 7. 自动 SLEEP 唤醒，自动退出 RX

控制位	值
SLEEP_TIMER_EN	1
RX_DC_EN	0
RX_TIMER_EN	1
RX_EXIT_STATE	1/2/5

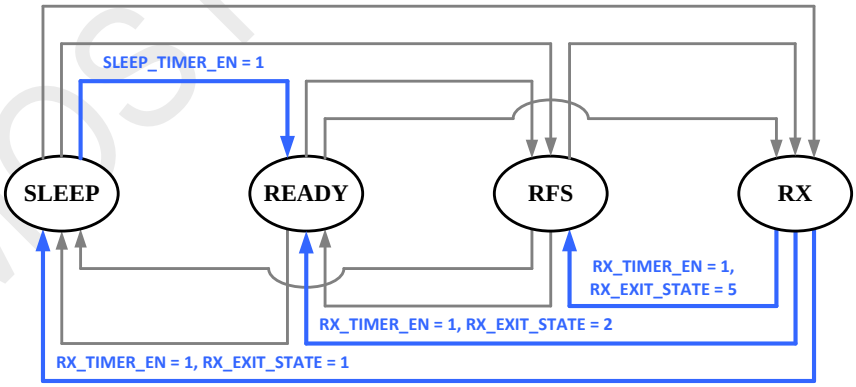


图 5. 接收机自动睡眠唤醒，自动退出 RX

1.2.5 全自动接收

全自动接收模式，就是一旦开始运转，完全不需要（也不能）由 MCU 参与切换状态。MCU 只能通过预先设置好的中断来获取芯片的工作状态，并进行需要的操作。需注意的是，在进入全自动发射之前，MCU 必须在 READY 状态将包格式，FIFO 的工作模式，以及中断和 IO 配置好。

表 8. 全自动接收

控制位	值
SLEEP_TIMER_EN	1
RX_DC_EN	1
RX_TIMER_EN	1
RX_EXIT_STATE	1

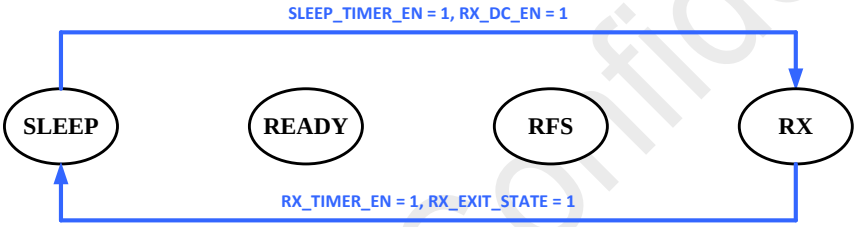


图 6.接收机全自动控制

这种模式的限制是，MCU 的速度要足够快，必须利用好中断和芯片交互，在 RX 的时候就读取完 FIFO。

对 MCU 与 CMT2310A 的配合，首先要注意的是遵守操作规则，一旦开启了自动控制，就不能手动发命令去做同样的状态切换。其次需特别注意的是，如果开启了自动退出 RX，并且退出状态是 SLEEP 的话，那么退出到 SLEEP 之后，**位于控制区 2 的中断状态会全部丢失**，因此，如果使用这种模式，一定要全面考虑，让 MCU 能够稳妥地与芯片交互，不会出现等不到中断的情况。通常我们会建议用户将 RX_EXIT_STATE 设置为 2，即自动退出 RX 后停留在 READY，这时所有中断状态会保留，功耗会大大降低，也可以让 MCU 安全地处理完所有工作，然后再手动切换到 SLEEP。

1.3 TX 的 Duty-Cycle 模式

相关寄存器及说明如下。

表 9. Tx Duty-Cycle 相关寄存器

寄存器名	位数	R/W	比特名	功能说明
CTL_REG_7 (0x07)	4	RW	API_STOP	用户配置该比特用于停止 feature 功能，置 1 后并不是立刻生效，而是在芯片下一次切换状态时生效。feature 成功停止后返回到 Ready 状态。 0：不停止 feature

寄存器名	位数	R/W	比特名	功能说明
				1：停止 feature
CTL_REG_23 (0x17)	7	RW	API_DONE_EN	api_done_flag 和 api_done 中断使能配置位，使能之后，api_done_flag 对应事件有效时，api_done_flag 会置 1，也会产生 api_done 中断。 0：不使能 1：使能
	4	RW	TX_DC_MAX_EN	tx_dc_max_flag 和 tx_dc_max 中断使能配置位，使能之后，tx_dc_max_flag 对应事件有效时，tx_dc_max_flag 会置 1，也会产生 tx_dc_max 中断。 1：使能 0：不使能
CTL_REG_31 (0x1F)	7	W	API_DONE_CLR	api_done_flag 清 0 位。 0：NA 1：将 api_done_flag 清 0
	4	W	TX_DC_MAX_CLR	tx_dc_max_flag 清 0 位。 1：将 tx_dc_max_flag 清 0 0：对 tx_dc_max_flag 没有效果
CTL_REG_32 (0x20)	7	R	API_DONE_FLAG	api_done_flag 标志位，当 api_stop 比特被置 1 并被成功响应后，芯片返回到 READY 状态，芯片在 api_done_en 使能的情况下会将 api_done_flag 标志位置 1。 0：api_stop 未成功响应 1：api_stop 成功响应，停止 feature
	4	R	TX_DC_MAX_FLAG	如果 tx duty clcle 没有使能 persistent 模式，当达到了最大发射次数，并且 tx_dc_max_en 使能的情况下，会将 tx_dc_max_flag 置 1。 1：tx duty cycle 达到最大发射次数 0：tx duty cycle 未达到最大发射次数
CTL_REG_96 (0x60)	6:4	RW	TX_EXIT_STATE<2:0>	完成发射后自动退出到设定的状态，只在 packet 模式下有效，否则芯片不会自动退出 TX 状态，而是等待 MCU 发 go_*命令来切换。 1：SLEEP 2：READY 3：TFS 4：TX 5：RFS 6：RX Others：SLEEP
	1	RW	TX_DC_PERSIST_EN	TX 模式下，duty cycle 发射的配置。 0：完成 TX_DC_TIMES 配置的次数就退出 1：一直进行，直到这个比特配置为 0
	0	RW	TX_DC_EN	TX Duty Cycle 的使能。

寄存器名	位数	R/W	比特名	功能说明
				0：不使能 1：使能
CTL_REG_110 (0x6E)	7:0	RW	TX_DC_TIMES<7:0>	TX Duty Cycle 模式下，非 persistent 模式下，规定的最大发射次数
CTL_REG_112 (0x70)	7:0	R	TX_DC_DONE_TIMES<7:0>	TX Duty Cycle 模式下已经完成的发射次数

要控制 TX 的 Duty-Cycle 模式，共需要使用 3 个寄存器控制位。对于 TX 来说，没有 TX TIMER，因为发射是主动行为，而功率非常大，在 packet 模式下完成发射之后，芯片默认的操作就是自动退出 TX 状态，因此不需要特定的计数器来计数，发射时间完全根据要发射的内容而定。

下面结合实际的应用需求，下面列出这 3 个控制位的 3 种组合，以及对应的运转模式。控制原则与 RX 的 Duty-Cycle 模式相同。

1.3.1 自动退出 TX

此模式为最简单的模式，就是完成 TX 后根据不同的 TX_EXIT_STATE 自动切换到对应的状态，其余情况都需由 MCU 手动控制切换。

表 10. 自动退出 TX

控制位	值
SLEEP_TIMER_EN	0
TX_DC_EN	0
TX_EXIT_STATE	1/2/3

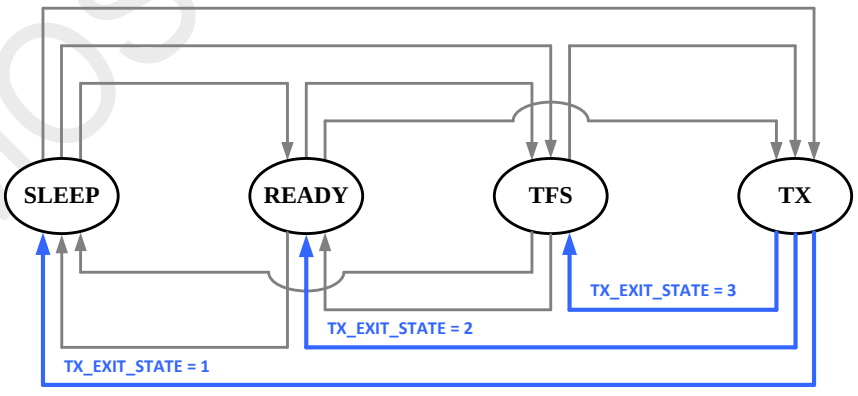


图 7. 发射机自动退出发射

1.3.2 自动 SLEEP 唤醒，自动退出 TX

这种模式下，除了保持自动退出 TX，还打开了 SLEEP TIMER，自动唤醒之后，芯片会切换到 READY，等待 MCU 操作。

表 11. 自动 SLEEP 唤醒，自动退出 TX

控制位	值
SLEEP_TIMER_EN	1
TX_DC_EN	0
TX_EXIT_STATE	1/2/3

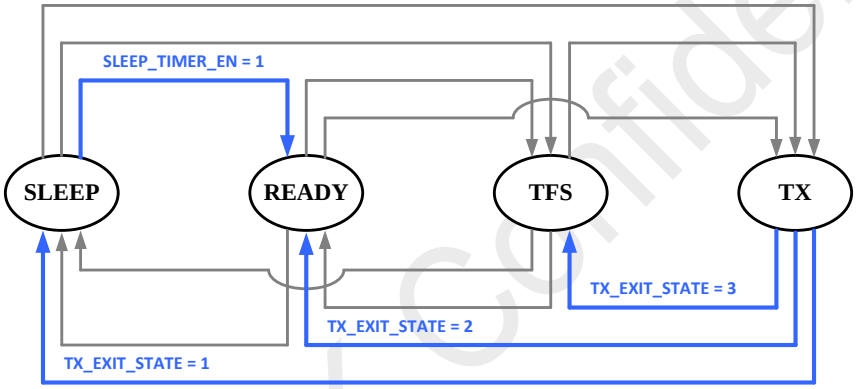


图 8. 发射机自动睡眠唤醒，自动退出发射

1.3.3 全自动发射

全自动发射模式，此模式一旦开始运转，完全不需要（也不能）MCU 参与切换状态。MCU 只能通过预先设置好的中断来获取芯片的工作状态，并进行需要的操作。需注意的是，在进入全自动发射之前，MCU 必须在 READY 将要发射的数据先填好，并将包格式，FIFO 的工作模式，以及中断和 IO 配置好。

表 12. 全自动发射

控制位	值
SLEEP_TIMER_EN	1
TX_DC_EN	1
TX_EXIT_STATE	1

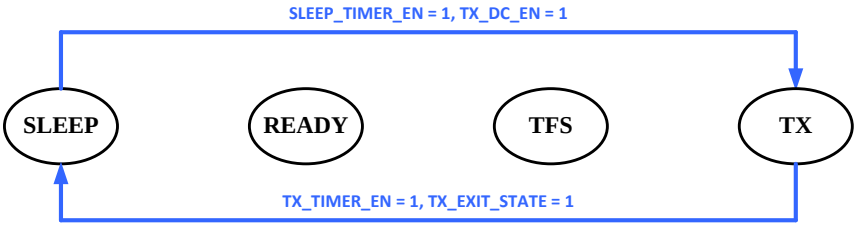


图 9. 发射机全自动发射

1.4 进入和退出 Duty-Cycle 模式

1.4.1 进入 Duty-Cycle 模式

芯片初始化完成后，配置阶段就可以通过配置相关的寄存器进入想要的 Duty-Cycle 模式，必须在 READY 完成配置并手动进入 SLEEP 后，芯片才会正式开始按照配置来进行运转。

1.4.2 退出 Duty-Cycle 模式

对于非全自动的 Duty-Cycle 模式，系统总会停在某个状态等待 MCU 操作，MCU 可以将系统切换回 READY，然后按照之前介绍的配置流程重新配置一次相关的几个寄存器，就可以退出 Duty-Cycle 模式了。

对于全自动的 Duty-Cycle 模式，无论是 TX 还是 RX 的，MCU 并不能准确地知道芯片的运转状态，所以要有一个 100%可靠的机制去让 MCU 让芯片停止自动运行且切换回手动控制模式。

假设 CMT2310A 在上面介绍的初始化配置之后，就开始了全自动的 Duty-Cycle 运转，MCU 要操作下面的寄存器位才能安全退出：

1. 将 API_STOP 设置为 1，这时芯片会在下一次切换状态时返回到 READY 状态。
2. MCU 可以通过扫描 CTL_REG_MODE<7:0>寄存器，直到确认芯片进入了 READY。
3. 重新配置 Duty-Cycle 相关寄存器关掉全自动模式（同时也可以配置其它的寄存器），完成配置后，将 API_STOP 设置为 0。
4. 发送 go_sleep 命令让配置生效，这时系统会停留在 SLEEP 等待 MCU 继续操作。

2. 超低功耗（SLP）接收模式

CMT2310A 提供了一系列的选项，能够帮助用户在不同的应用需求下实现超低功耗（SLP - Supper Low Power）的接收。这些选项都必须在 RX_TIMER_EN 被设置为 1，即 RX 计时器有效的时候才会生效。SLP 接收的核心内容是如何让接收机在无信号的时候尽量缩短 RX 的时间，在有信号的时候又能够恰当地延长 RX 的时间进行接收，最终达到功耗最小化并稳定接收的效果。

2.1 SLP 接收相关的寄存器

对应的 RFPDK 的界面和参数如下。

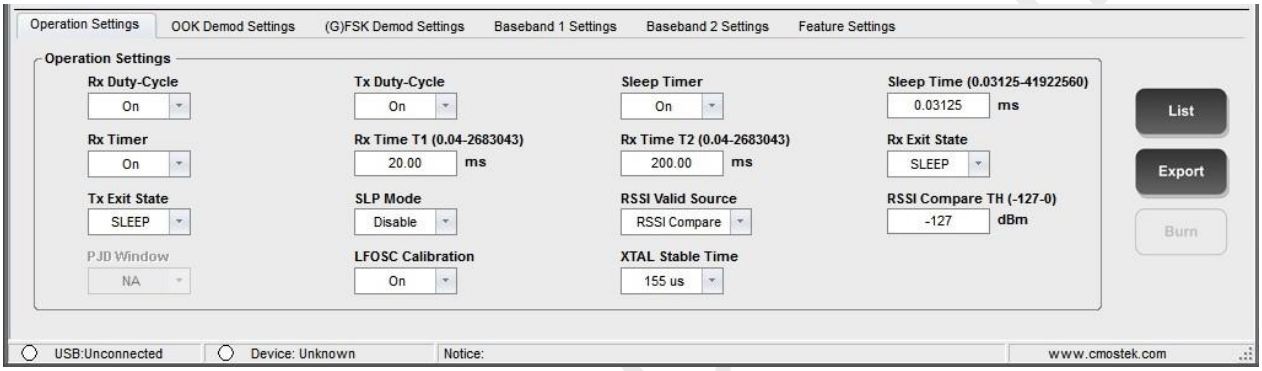


图 10. SLP 的 RFPDK 界面

表 13. SLP 相关参数

寄存器比特 RFPDK 参数	寄存器比特
此比特固定为 1	RX_AUTO_EXIT_DIS
SLP Mode	RX_EXTEND_MODE<3:0>

相关寄存器及说明如下表。

表 14. SLP 相关寄存器

寄存器名	位数	R/W	比特名	功能说明
CTL_REG_7 (0x07)	4	RW	API_STOP	用户配置该比特用于停止 feature 功能，置 1 后并不是立刻生效，而是在芯片下一次切换状态时生效。feature 成功停止后返回到 Ready 状态。 0：不停止 feature 1：停止 feature
CTL_REG_23 (0x17)	7	RW	API_DONE_EN	api_done_flag 和 api_done 中断使能配置位，使能之后，api_done_flag 对应事件有效时，api_done_flag 会置 1，也会产生 api_done

寄存器名	位数	R/W	比特名	功能说明
				中断。 0：不使能 1：使能
CTL_REG_31 (0x1F)	7	W	API_DONE_CLR	api_done_flag 清 0 位。 0：NA 1：将 api_done_flag 清 0
CTL_REG_32 (0x20)	7	R	API_DONE_FLAG	api_done_flag 标志位，当 api_stop 比特被置 1 并被成功响应后，芯片返回到 READY 状态，芯片在 api_done_en 使能的情况下会将 api_done_flag 标志位置 1。 0：api_stop 未成功响应 1：api_stop 成功响应，停止 feature
CTL_REG_97 (0x61)	6:4	RW	RX_EXIT_STATE <2:0>	rxtimer timeout 之后返回的状态。 1：SLEEP 2：READY 3：TFS 4：TX 5：RFS 6：RX Others：SLEEP
	3	RW	TIMER_RX_EN	rx timer 使能位。 0：不使能 1：使能
	0	RW	RX_DC_EN	RX Duty Cycle 的使能。 0：不使能 1：使能
CTL_REG_98 (0x62)	7	RW	PKT_DONE_EXIT_EN	芯片成功收到 packet 产生 pkt_done 时是保持当前状态还是立即退出返回到 RX_EXIT_STATE 对应的状态。 0：芯片保持当前状态 1：芯片根据 RX_EXIT_STATE 配置返回到对应状态
	3:0	RW	SLP_MODE<3:0>	RX Duty Cycle 模式配置。 有效范围 0-13。
CTL_REG_99 (0x63)	7:0	RW	SLEEP_TIMER_M<7:0>	定义了 SLEEP TIMER 的计时时间，公式如下： T = M x 2^(R+1) x 31.25 us R 的取值范围是 0-26。
CTL_REG_100 (0x64)	7:5	RW	SLEEP_TIMER_M<10:8>	
	4:0	RW	SLEEP_TIMER_R<4:0>	
CTL_REG_101 (0x65)	7:0	RW	RX_TIMER_T1_M<7:0>	定义了 RX T1 TIMER 的计时时间，公式如下： T = M x 2^(R+1) x 20 us R 的取值范围是 0-21
CTL_REG_102 (0x66)	7:5	RW	RX_TIMER_T1_M<10:8>	
	4:0	RW	RX_TIMER_T1_R<4:0>	
CTL_REG_103	7:0	RW	RX_TIMER_T2_M<7:0>	定义了 RX T1 TIMER 的计时时间，公式如下：

寄存器名	位数	R/W	比特名	功能说明
(0x67)				T = M x 2^(R+1) x 20 us R 的取值范围是 0-21。
CTL_REG_104 (0x68)	7:5	RW	RX_TIMER_T2_M<10:8>	
	4:0	RW	RX_TIMER_T2_R<4:0>	
CTL_REG_105 (0x69)	3	RW	TIMER_SLEEP_EN	SLEEP TIMER 的使能。 0：不使能 1：使能
CTL_REG_106 (0x6A)	3:2	RW	CCA_INT_SEL<1:0>	rssi_pjd_valid 产生条件配置。 00：pjd_ok 01：rssi_ok 10：pjd_ok & rssi_ok 11：无效

2.2 低功耗收发基本原理

传统的短距离无线收发系统，一般都会以下面这种基本的方案实现低功耗收发。CMT2310A 同样兼容这种方案，并且在这个基础上扩展出 13 种更加节省功耗的方案。下面先介绍最基本的方案，即将 SLP_MODE <3:0>设置为 0 时就可以实现的方案：

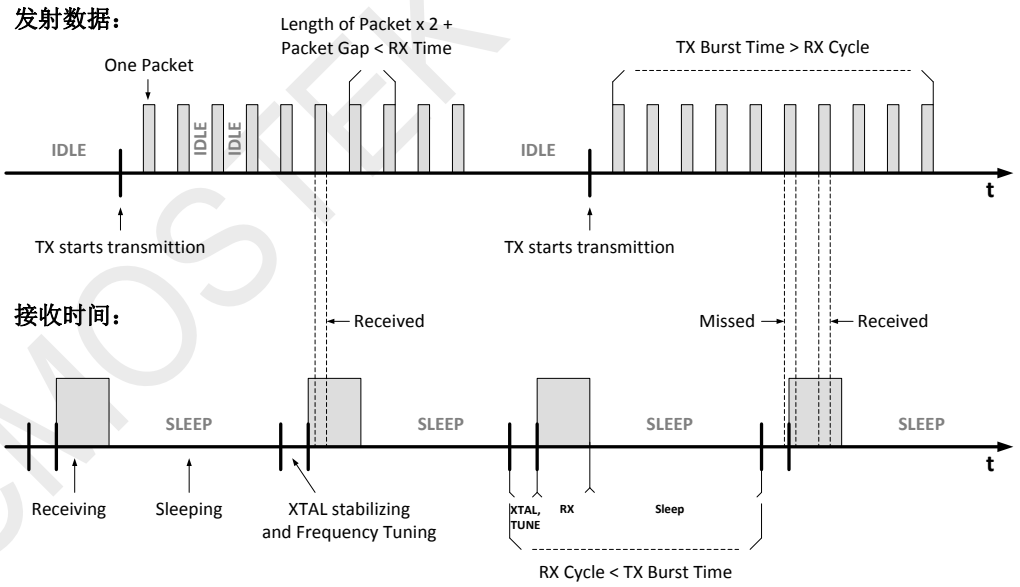


图 11. 基本的低功耗收发方案

从图中可以看出，只要满足两个计算关系，就可以做到 RX 在 Duty-Cycle 接收模式下，一定能够捕捉到 TX 的数据：

1. 完整的 RX 周期< TX 每次批量发送 N 个数据包的总时间
2. RX 时间> 2 个数据包加上 1 个包间隔的时间

其中，一个完整的 RX 的周期 = RX 时间 + 睡眠时间 + 晶体起震和稳定时间 + PLL 频率校正时间。

可见，使用这种基本的低功耗方案，受到计算关系的约束，用户首先需要在 SLEEP 的时间和发射数据长度之间做出折衷，即 RX 省电一点，还是 TX 省电一点；第二，用户必须将 RX 的时间窗口设置得足够大，才能够 100%捕捉到数据。

2.3 信道侦听

在介绍各种 SLP 模式之前，先介绍 SLP 的重要辅助机制 – 信道侦听。此机制通过监听有效信号是否出现，从而产生信号 RSSI_PJD_VLD（1 表示信号出现，0 表示噪声），此信号不仅会作为中断输出到 GPIO，而且会作为一个触发条件，辅助 SLP 的实现。

信道侦听的机制有两种，分别是相位跳变检测（PJD – Phase Jump Detector）和 RSSI 对比。

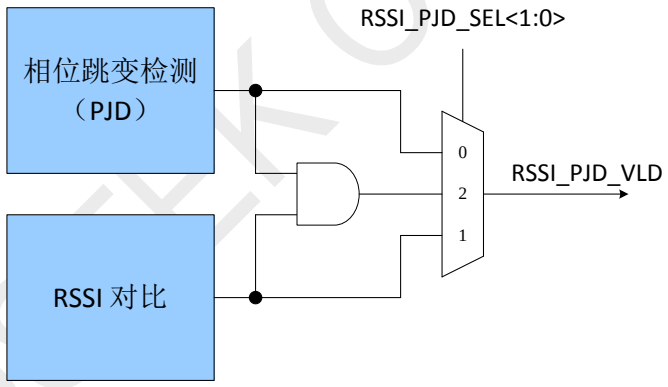


图 12. 信道侦听的机制

需要注意的是，PJD 只有在 2-FSK 模式下才可使用；RSSI 对比在 2-FSK，4-FSK 和 OOK 模式下都可使用。

2.3.1 信道侦听相关寄存器

对应的 RFPDK 的界面和参数如下。

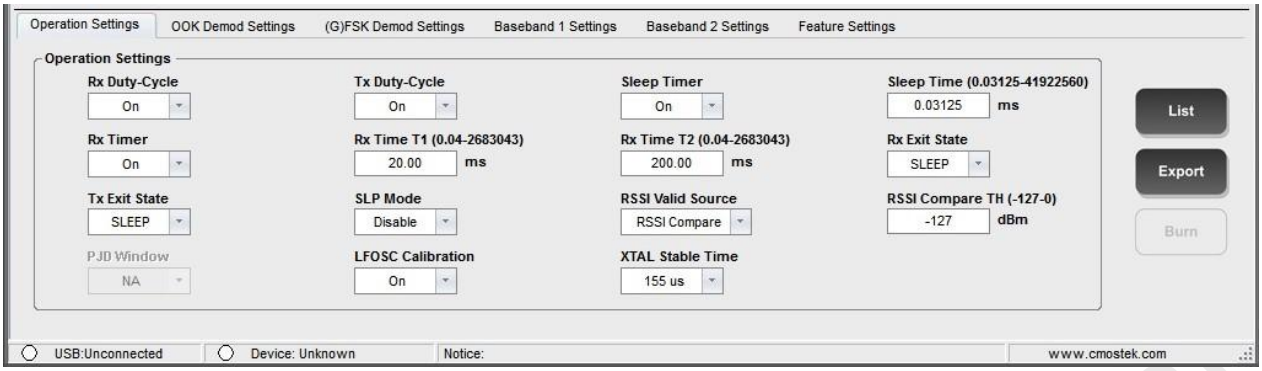


图 13. 信道侦听的 RFPDK 界面

表 15. 信道侦听相关参数

寄存器比特 RFPDK 参数	寄存器比特
RSSI PJD Valid Source	CCA_INT_SEL<1:0>
PJD Window	此参数定义了 PJD 需要检测多少次跳变才判断进来的是噪声还是信号。 0：4 次 1：6 次 2：8 次 3: 10 次

寄存器的内容和描述详见下表。

表 16. 位于配置区的寄存器

寄存器名	位数	R/W	比特名	功能说明
CTL_REG_106 (0x6A)	3:2	RW	RSSI_PJD_SEL<1:0>	RSSI_PJD_VALID 中断产生条件： 00：PJD 有效 01：RSSI 有效 10：PJD 和 RSSI 都有效 11：NA

2.3.2 RSSI 对比

此功能会在后面的章节“RSSI 测量与对比”中详细介绍，此处只做概要介绍。进行 RSSI 对比的原理是，信号或者噪声的 RSSI 比阈值高，RSSI_VLD 就会有效，否则无效。这种方式的优点是无论是 FSK 和 OOK 都可用，缺点是阈值的设置需要根据实际应用环境进行调试，要尽量避免被噪声和干扰信号触发。用这种方法产生 RSSI_VLD 辅助 SLP 优势并不明显，因此我们主要关注下面的 PJD 方法。

2.3.3 相位跳变检测（PJD）

PJD 是一种新的技术，在芯片进行 FSK 解调的时候，可用通过观察接收信号的跳变特性，来决定进来的是噪声还是有用信号。PJD 认为输入信号从 0 到 1 或者从 1 到 0 切换就是一次相位跳变，用户仅仅需要去配置 PJD_WIN_SEL<1:0>，来告诉 PJD 需要检测多少次信号跳变才能输出判断结果。

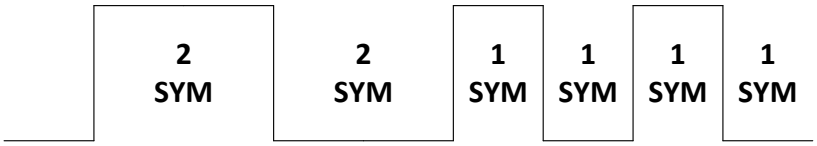


图 14.接收信号跳变图

如果上图所示，共接收了 8 个 symbol，但是跳变只出现了 6 次，因此跳变数并不能等同于 symbol 数量。只有在接收 preamble 时，跳变数才等同于 symbol 数。用户设置的时候要注意这一点。

后面的章节会介绍，RSSI_VLD 信号是如何辅助实现超低功耗（SLP）接收模式的。总的来说，PJD 跳变次数越多，判断结果越可靠；越少，就越快完成。如果接收的时间窗口很小，那么就需要将检测次数减少来满足窗口设置的要求。

根据测试得出的数据，一般来说，跳变次数是 4 次就已经可以达到非常可靠的检测效果，即不会将噪声误判为有用信号，有用信号来的时候不会检测不到。

2.4 SLP 接收模式详解

如上面介绍，SLP 的核心内容是控制好 RX 的时间，达到一个目的：平时没有有用信号的时候，RX 的时间非常短，只用于检测有用信号是否到来；当有用信号到来的时候，RX 的时间会延长，可以成功接收需要的数据包。

所以，SLP 是在前面介绍的各种手动、半自动、全自动的 RX Duty-Cycle 控制模式的基础上，对 RX 状态的进一步控制。即 RX_EXTEND_MODE<3:0>定义的 14 种 SLP 模式，全部都是为了控制 RX 的时间，除了 RX 状态以外的控制无论是自动还是手动，与 SLP 本身没有必然的联系，用户可以在不同的 RX Duty-Cycle 模式下实现不同的 SLP 模式。

下文将详细解释这 14 种 SLP 模式。首先，我们假设 CMT2310A 是工作在 RX Duty-Cycle 模式下，自动 SLEEP 唤醒，自动退出 RX 并切换到 READY，MCU 负责从 READY 切换到 SLEEP，和从 READY 切换到 RX。所有 13 种 SLP 模式，都是在这个基础上演变而来。

表 17. SLP 详解例子中的 RX Duty-Cycle 模式

控制位	值
SLEEP_TIMER_EN	1
RX_DC_EN	0
RX_TIMER_EN	1
RX_EXIT_STATE	2

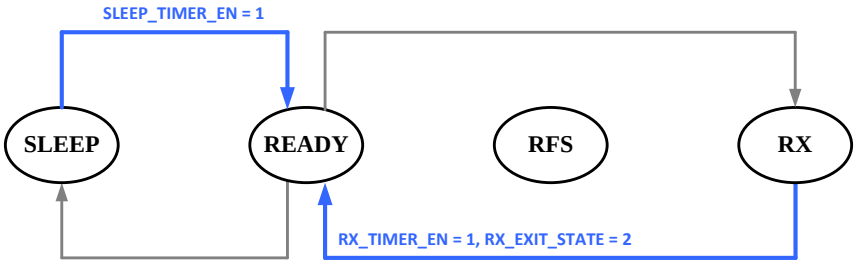


图 15. SLP 详解例子中的 RX Duty-Cycle 模式

我们会以在这种 RX Duty-Cycle 模式为基础，来介绍各种 SLP 模式。以下是 14 种模式的汇总，表中提到的 T1 和 T2 分别指可用寄存器进行设定的 RX T1 和 T2 时间窗口。

表 18. SLP 的 14 种模式

编号	RX 的延长方式	RX 的延长条件
0	如果配置为 0，不做任何延长，T1 计时结束即离开 RX。	无
1	T1 内一旦满足检测条件，即离开 T1，将控制权交给 MCU。	RSSI_VLD 有效
2		PREAM_OK 有效
3		RSSI_VLD 与 PREAM_OK 同时有效
4	T1 内只要检测到 RSSI 有效，即退出 T1 并一直处于 RX，直到 RSSI 不满足就退出 RX。	RSSI_VLD 有效
5	T1 内一旦满足检测条件，即切换到 T2，T2 计时结束后即退出 RX。	RSSI_VLD 有效
6		PREAM_OK 有效
7		RSSI_VLD 与 PREAM_OK 同时有效
8		PREAM_OK 或 SYNC_OK 任意一个有效
9		PREAM_OK 或 NODE_OK 任意一个有效
10		PREAM_OK 或 SYNC_OK 或 NODE_OK 任意一个有效
11	T1 内一旦满足检测条件，即切换到 T2，T2 内一旦检测到 SYNC 即退出 T2 并将控制权交给 MCU，否则 T2 计时结束后就退出 RX。	RSSI_VLD 有效
12		PREAM_OK 有效
13		RSSI_VLD 与 PREAM_OK 同时有效

上面介绍的 14 种模式中，有几种都是使用 RSSI_VLD 作为触发条件的。使用由 PJD 产生的 RSSI_VLD 辅助超低功耗接收，是 CMT2310A 产品的一个创新，效果显著，推荐用户使用。而选项 3 结合了 PJD 产生的 RSSI_VLD 和 PREAM_OK 来做判断条件，增加了可靠性的同时也不会增加太多的时间。

如果使用选项 3，假设发射机会发送足够长的 preamble，T1 可以使用下面的方法设置：

预留 8 个 symbol 给接收机做 AFC，将 PJD 的跳变数设为 6，将 preamble 的长度设为 4 或 8 个 symbol，然后增加 6-8 个 symbol 的时间用于探测 RSSI_VLD& PREAM_OK 的延长条件成立。所以 T1 的时间共为 14 – 16 个 symbol。如果发射接收双方的晶体频率偏差不大，例如远小于设置的 deviation，那么前面 8 个 symbol 的 AFC 时间可以相对减少，需要用户经过实际测试来确认。

2.4.1 SLP 模式 0

当模式设为 0 的时候，接收时间等于 RX T1，不做任何延长处理。从下面的示意图中可见，MCU 只负责发送 go_rx 和 go_sleep 命令切换状态，CMT2310A 会进行自动睡眠唤醒和自动退出 RX，并产生相应的中断。

图中给出的 RX 和 SLEEP 等时间只是用于举例说明，无特殊意义。

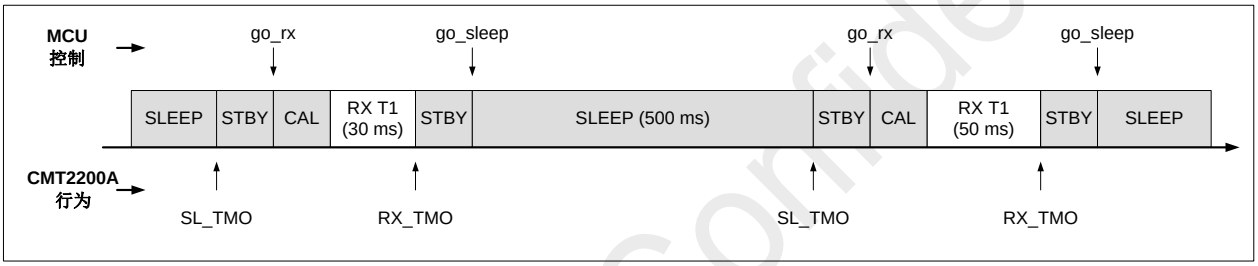


图 16. SLP 模式 0

2.4.2 SLP 模式 1-3

当模式设置成 1-3 的时候，RX T1 内一旦满足检测条件，RX T1 就停止计时，芯片停留在 RX，并将控制权交给 MCU；否则 RX T1 计时结束后即退出 RX。三种不同的条件如下：

1：检测条件为 RSSI_VLD 有效
2：检测条件为 PREAM_OK 有效
3：检测条件为 RSSI_VLD 与 PREAM_OK 同时有效

下面以选项 2 为例，给出 TX，MCU 和 RX 配合工作的时序图：

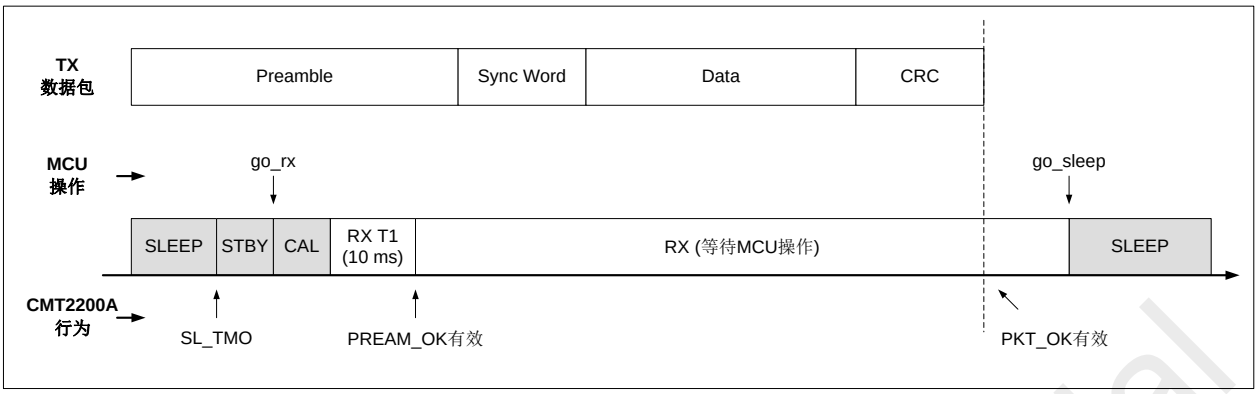


图 17. SLP 模式 1-3

需注意的是，RX T1 的预设时间是 10 ms，但是一旦检测到条件满足时就停止了，例如会停止在 7.5 ms，然后下一次重新进入 RX 时才会复位并重新计时。

以下这些场景比较适合使用模式 1 到 3：

在某些应用中，TX 发射的内容变化比较大，即每次传送的内容，包长度，包个数都不定。这样的情况下，当检测条件满足，RX T1 停止之后，将 RX 的切换权交回给 MCU，由 MCU 自己决定还需要接收多长时间会比较好。

以下为 3 种条件的特点比较：

选项 1 的检测条件 RSSI_VLD 是信道侦听的输出结果。如前面介绍，信道侦听在 PJD 的辅助下，能够对是否有有用信号出现，做出非常快速并可靠的判断，更重要的是，它不需要受数据格式的限制，不需要 TX 发送很长的 Preamble，等。因此选项 1 的好处就是，当 RX T1 的窗口设置到很小（例如 5-8 个 symbol）的时候，仍然能够非常可靠地进行检测判断。

选项 2 的检测条件是 PREAM_OK，是 Packet 产生的中断。这是一个传统的检测条件，一般来说，需要至少 2 个 byte 以上的 Preamble 长度才能够保证检测是可靠的，所以它的好处是直观易理解，缺点是需要包格式里面一定要有 Preamble 的存在，而且 TX 发射的 Preamble 的长度必须足够长（覆盖 2 个 RX T1 和 1 个 SLEEP 时间），RX T1 也要覆盖 2 个 byte 甚至是更长的 Preamble 才能实现可靠检测。

选项 3 的检测条件是两者同时有效，意味着这个检测条件就比较苛刻，极不容易被误触发，具有非常高的可靠性。CMT2310A 的 RX_PREAM_SIZE 的最小值可以设置为 4 个 symbol，如果结合 RSSI_VLD 信号，既不会增加 RX T1 的时间，也能够达到更加可靠的检测效果。

2.4.3 SLP 模式 4

当模式设置为 4 的时候，RX T1 内只要检测到 RSSI_VLD 有效，RX T1 就停止计时，并一直处于 RX，直到 RSSI_VLD 无效才自动退出 RX；RX T1 内若检测不到 RSSI_VLD 有效，计时结束后退出 RX。

图 18.SLP 模式 4

2.4.4 SLP 模式 5-10

当模式设置为 5 ~ 10 时，RX T1 内一旦满足检测条件，就切换到 RX T2，RX T2 计时结束后才自动退出 RX；否则 RX T1 之内检测不到条件满足，计时结束后就退出 RX。六种不同的条件如下：

5：检测条件为 RSSI_VLD 有效
6：检测条件为 PREAM_OK 有效
7：检测条件为 RSSI_VLD 与 PREAM_OK 同时有效
8：检测条件为 PREAM_OK 或 SYNC_OK 任意一个有效
9：检测条件为 PREAM_OK 或 NODE_OK 任意一个有效
10：检测条件为 PREAM_OK 或 SYNC_OK 或 NODE_OK 任意一个有效

下面以选项 6 为例，给出 TX，MCU 和 RX 配合工作的时序图：

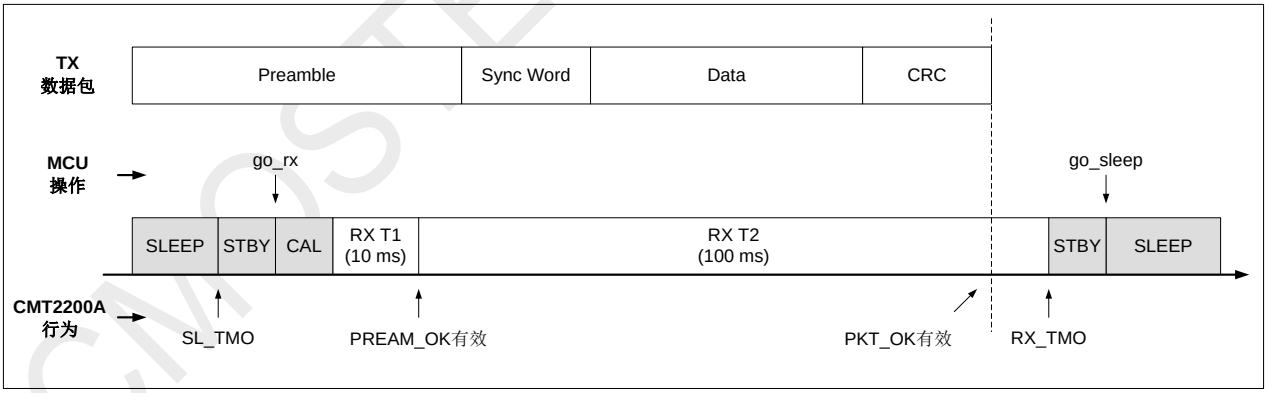


图 19. SLP 模式 6

用户需要将 RX T2 设置得足够长，保证可以完整地接收所有需要的内容。

以下这些场景比较适合使用模式 5 到 10：

相比起模式 1-3，模式 5-10 的改变之处在于 RX T2 的存在。芯片切换到 RX T2 之后，不需要 MCU 参与控制，也不需要 MCU 自己做超时，都会在 RX T2 超时后自动退出 RX。因此这些模式比较适合每次发送的数据长度都比较相近的应用，这样 RX T2 的时间就比较好设置。

以下是模式 8 到 10 的实用意义：

在一些应用中，用户为了尽量降低 TX/RX 的功耗，其低功耗收发方案就不会符合 9.2 章节介绍的计算原则，例如会将 RX 时间设置得比较短，只能有概率地捕捉到 TX 发送过来的检测条件，而无法实现 100%捕捉。将检测条件设为 2 个或 3 个条件任意一个有效，就会提高成功捕捉几率，但同时可靠性也会减低。如果经过市场验证，使用有 PJD 辅助的 RSSI_VLD 能够可靠地工作，那么选项 8-10 其实没有实用意义。

包格式里面如果没有 NODE ID，也可以 9 和 10：

在包里面没有 NODE ID 的时候，将包格式相关的寄存器 NODE_FREE_EN 设置为 1，意思就是 NODE_VALUE 这个寄存器不再用于定义 NODE ID，而是根据不同的 NODE_SIZE 来自定义一个码值，接收机只要检测到数据中有符合这个码值的，就会产生 NODE_OK 中断，因此条件 9 和 10 也可以被满足。

2.4.5 SLP 模式 11-13

当模式设置成 11-13 的时候，RX T1 内一旦满足检测条件，就切换到 RX T2，RX T2 内一旦检测到 SYNC_OK，RX T2 就停止计时，芯片停留在 RX，并将控制权交给 MCU；否则在 RX T1 或者 RX T2 之内检测不到条件满足，计时结束后就退出 RX。三个条件如下：

11：检测条件为 RSSI_VLD 有效
12：检测条件为 PREAM_OK 有效
13：检测条件为 RSSI_VLD 与 PREAM_OK 同时有效

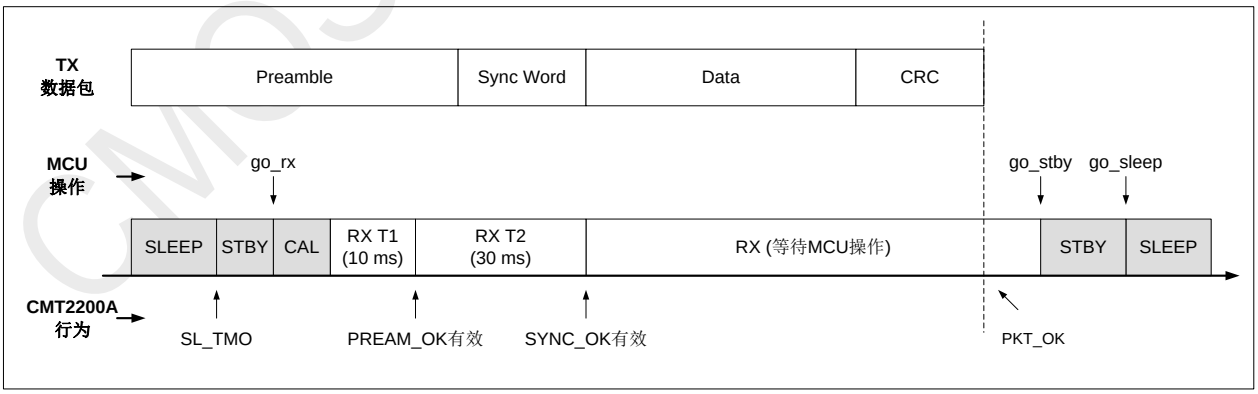


图 20.SLP 模式 11-13

以下这些场景比较适合使用模式 11 到 13：

对于数据包有 Preamble，Sync Word 的应用，都可以使用这 3 种模式。这 3 种模式的好处是，如果在同一个频道中有不同的发射数据，那么光靠检测 Preamble 或者 RSSI 还不能说明接收到的数据是想要的，加上必须满足固定的 Sync Word 作为第二级的检测条件，就会非常可靠。如果第二级检测不满足，就会在 RX T2 之后就退出了，这样也不会由于误触发而浪费过多的功耗。

3. 自动跳频接收（RX Auto Hop）

3.1 RX Auto Hop 模式相关的寄存器

寄存器的内容和描述详见下表。

表 19. 位于配置区的寄存器

寄存器名	位数	R/W	比特名	功能说明
CTL_REG_7 (0x07)	4	RW	API_STOP	用户配置该比特用于停止 feature 功能，置 1 后并不是立刻生效，而是在芯片下一次切换状态时生效。feature 成功停止后返回到 Ready 状态。 0：不停止 feature 1：停止 feature
CTL_REG_23 (0x17)	7	RW	API_DONE_EN	api_done_flag 和 api_done 中断使能配置位，使能之后，api_done_flag 对应事件有效时，api_done_flag 会置 1，也会产生 api_done 中断。 0：不使能 1：使能
CTL_REG_31 (0x1F)	7	RW	API_DONE_CLR	api_done_flag 清 0 位 0：NA 1：将 api_done_flag 清 0
CTL_REG_32 (0x20)	7	R	API_DONE_FLAG	api_done_flag 标志位，当 api_stop 比特被置 1 并被成功响应后，芯片返回到 READY 状态，芯片在 api_done_en 使能的情况下会将 api_done_flag 标志位置 1 0：api_stop 未成功响应 1：api_stop 成功响应，停止 feature
CTL_REG_11 (0x0B)	7:0	RW	FREQ_DONE_TIMES	rx_auto_hop 已经跳频完成的次数，当 rx_auto_hop 使能并完成了一次跳频后，该值会自动加 1。用户可以配置该值，结合 freq_times 来实现实际需要的跳频次数；当用户配置该值大于 64 时，芯片认为配置的是 0。
CTL_REG_13 (0x0D)	7:0	RW	FREQ_TIMES	配置 rx_auto_hop 跳频表的大小，跳频表最大深度为 64，所以当配置值大于 64 时，认为配置的是最大深度 64。
CTL_REG_12 (0x0C)	7:0	RW	FREQ_SPACE	自动跳频的频道间隔。
CTL_REG_14 (0x0E)	1	RW	RX_HOP_PERSIST	该比特起作用于 rx_auto_hop 跳完整个表之后仍未找到信号的时候，该比特决定是从表头开始继续跳频扫描还是根据 rx_timeout_state 回到对应的状态。 1：从表头开始继续跳频扫描

寄存器名	位数	R/W	比特名	功能说明
				0：回到 rx_timeout_state 对应的状态
	0	RW	FREQ_SW_STATE	该比特起作用于 rx_auto_hop 在经过当前信道窗口仍未找到信号的时候，该比特决定是回到 READY 还是 RFS 状态下配置跳频表里的下一跳频信道，然后进入 RX 继续扫描信号。 1：回到 RFS 信道配置下一信道 0：回到 READY 信道配置下一信道
CTL_REG_37 (0x25)	7:0	R	REQ_CHANL_ACT	rx_auto_hop 模式时，当前使用的频道值，提供给用户做进一步操作。
CTL_REG_97 (0x61)	6:4	RW	RX_EXIT_STATE[2:0]	rxtimer timeout 之后返回的状态。 1：SLEEP 2：READY 3：TFS 4：TX 5：RFS 6：RX Others：SLEEP
	3	RW	TIMER_RX_EN	rx timer 使能位。 0：不使能 1：使能
	1	RW	RX_AUTO_HOP_EN	Rx Auto Hop 功能使能位。 0：不使能 1：使能
CTL_REG_98 (0x62)	7	RW	PKT_DONE_EXIT_EN	芯片成功收到 packet 产生 pkt_done 时是保持当前状态还是立即退出返回到 RX_EXIT_STATE 对应的状态。 0：芯片保持当前状态 1：芯片根据 RX_EXIT_STATE 配置返回到对应状态
	2:0	RW	RX_HOP_METHOD<3:0>	Rx Auto Hop 模式配置。 有效范围 0-6。
CTL_REG_101 (0x65)	7:0	RW	RX_TIMER_T1_M<7:0>	定义了 RX T1 TIMER 的计时时间，公式如下： T = M x 2^(R+1) x 20 us R 的取值范围是 0-21。
CTL_REG_102 (0x66)	7:5	RW	RX_TIMER_T1_M<10:8>	
	4:0	RW	RX_TIMER_T1_R<4:0>	
CTL_REG_103 (0x67)	7:0	RW	RX_TIMER_T2_M<7:0>	定义了 RX T1 TIMER 的计时时间，公式如下： T = M x 2^(R+1) x 20 us R 的取值范围是 0-21。
CTL_REG_104 (0x68)	7:5	RW	RX_TIMER_T2_M<10:8>	
	4:0	RW	RX_TIMER_T2_R<4:0>	
CTL_REG_106 (0x6A)	3:2	RW	CCA_INT_SEL<1:0>	RSSI_PJD_VALID 中断产生条件： 00：PJD 有效 01：RSSI 有效 10：PJD 和 RSSI 都有效 11：NA

3.2 功能使用说明

RX_Auto_Hop 模式共有 7 种模式，可以通过 RX_HOP_METHOD 配置。RX Auto Hop 中 FREQ_TIMES<7:0>用于配置跳频表的大小，表格最多包含 64 个字节的空间供用户存入需要配置的信道。跳频的过程如下。

芯片进入一个信道进行接收，在对应模式下，如果一直没有检测到成功的事件，那么芯片根据 FREQ_SW_STATE 返回到 READY 或者 RFS 状态，自动配置跳频表内下一个字节的信道值，然后重新进入 RX 状态进行接收。如果自动切换状态为 READY，那么重新进入 RX 之前会校准 PLL 频率，会消耗对应的校准时间；如果自动切换状态为 RFS，那么将不会校准 PLL 频率，切换时间会较短。如果跳频表里面任意两个频点之间的间隔超过 2MHz，建议将 FREQ_SW_STATE 配置为 READY 状态，因为有重新校准 PLL 的必要。

如果检测到成功的事件，就会停止自动跳频，等待外部 MCU 进行处理，或者切换到预先配置好的状态。当跳完整个表仍未检测到成功的事件，如果 RX_HOP_PERSIST 配置为 1，那芯片会继续从头开始跳表，如果 RX_HOP_PERSIST 配置为 0，那么就返回到 READY 状态，完成自动跳频接收的操作。

跳频的目标频点的计算公式是：

FREQ = 10K x FREQ_SPACE<7:0> x FREQ_CHANL<7:0>

其中 FREQ_SPACE<7:0>是一个独立的寄存器，FREQ_CHANL<7:0>则由 64 个寄存器进行配置，每个寄存器存放一个频道值。

RX_Auto_Hop 模式提供 REQ_CHANL_ACT 寄存器用于告诉用户当前使用的频道值。另外还有 FREQ_DONE_TIMES 寄存器，用于读取当前已经完成的跳频次数。

RX_Auto_Hop 模式自动过程涉及到使用 RX T1 TIMER 和 RX T2 TIMER，所以使用过程中必须将 TIMER_RX_EN 置 1。RX T1 TIMER 通过 TIMER_M_RX_T1 和 TIMER_R_RX_T1 配置，实际定时时间可以由公式计算得出；RX T2 TIMER 通过 TIMER_M_RX_T2 和 TIMER_R_RX_T2 配置，实际定时时间可以由公式计算得出。

RSSI_PJD_SEL 用于决定 RSSI_PJD_VLD 是由 RSSI 决定还是 PJD 决定，RX_EXIT_STATE 决定在接收退出时返回到哪个状态，PKT_DONE_EXIT_EN 决定在收到 PKT_DONE 时是否要退出返回到 RX_EXIT_STATE 配置的状态，还是保持在当前状态。

用户完成配置之后，将 RX_AUTO_HOP_EN 置 1 使能，然后发送 go_rx 命令就可以进入了 RX_Auto_Hop 模式，自动运行过程中如果触发停止事件，芯片会退出自动过程停止到固定的某个状态，如果用户想自行停止 RX_Auto_Hop 功能，可以将 API_STOP 置 1，芯片检测到该比特之后，会在下一次状态切换时将 API_DONE_FLAG 置 1，保持当前配置然后退出 RX_Auto_Hop 返回到 READY 状态。

在 RX Auto Hop 停止后，如果 RX_AUTO_HOP_EN 保持为 1，那么用户下次发送 go_rx 命令，芯片会再次进入自动跳频接收操作，起始频点是上一次跳频的结束频点的下一个频点。如果用户想从跳频表中的第一个频点开始跳频，就在发送 go_rx 命令前，将 FREQ_DONE_TIMES 寄存器清零即可。

RX Auto Hop 共有下面的 7 种工作模式：

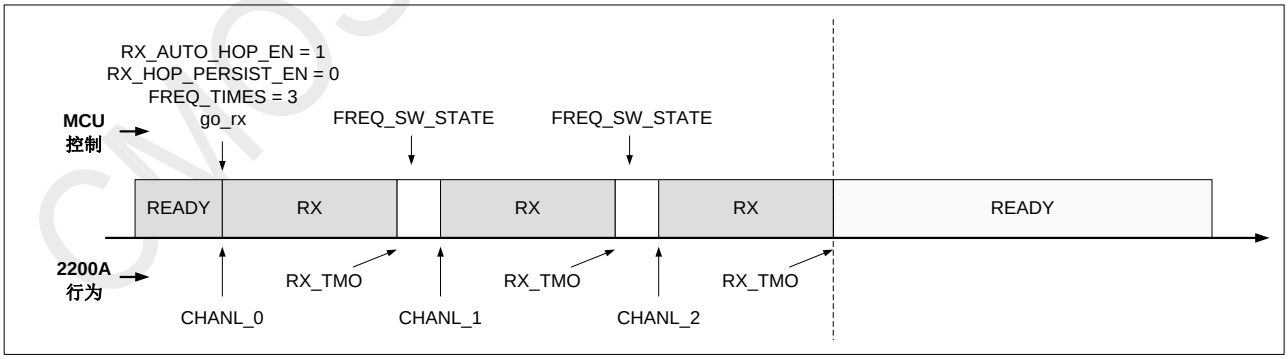
编号	RX 的延长方式	RX 的延长条件
0	如果配置为 0，不做任何延长。在 T1 超时前如果检测 PKT_DONE，并且 PKT_DONE_EXIT_EN = 1 退出自动跳频，把状态切换到 RX_EXIT_STATE；否则 T1 超时退出 RX，返回 FREQ_SW_STATE 状态，再继续跳往下一个频点。	无
1	T1 内一旦满足检测条件，即离开 T1，将控制权交给 MCU（保持状态）；如果检测不满足条件，即超时退出 RX，返回 FREQ_SW_STATE 寄存器指定的状态，继续跳往下一个频点。	RSSI_PJD_VLD 有效
2		PREAM_OK 有效
3		RSSI_PJD_VLD 与 PREAM_OK 同时有效
4	T1 内一旦满足检测条件，即切换到 T2；如果不满足条件溢出，就超时退出 RX，返回 FREQ_SW_STATE 寄存器指定的状态，继续跳往下一个频点。在 T2 内一旦检测到 SYNC 就退出 T2 并将控制权交给 MCU（保持状态），否则 T2 计时结束后即退出 RX，返回 FREQ_SW_STATE 寄存器指定的状态，继续跳往下一个频点。	RSSI_PJD_VLD 有效
5		PREAM_OK 有效
6		RSSI_PJD_VLD 与 PREAM_OK 同时有效

3.3 时序图说明

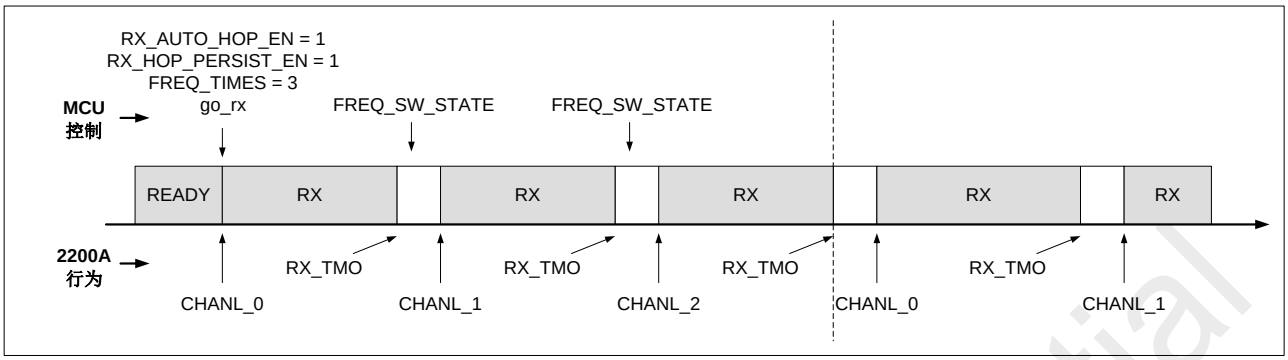
3.3.1 RX Auto Hop 跳表方式

当 RX Auto Hop 跳完整个表都没有检测成功时，有两种处理方式。当 RX_HOP_PERSIST_EN 为 0 时，芯片退出 RX Auto Hop 自动控制过程返回到 READY 状态；当 RX_HOP_PERSIST_EN 为 1 时，芯片重新从表头信道开始跳频检测信号。

1. RX_HOP_PERSIST_EN = 0



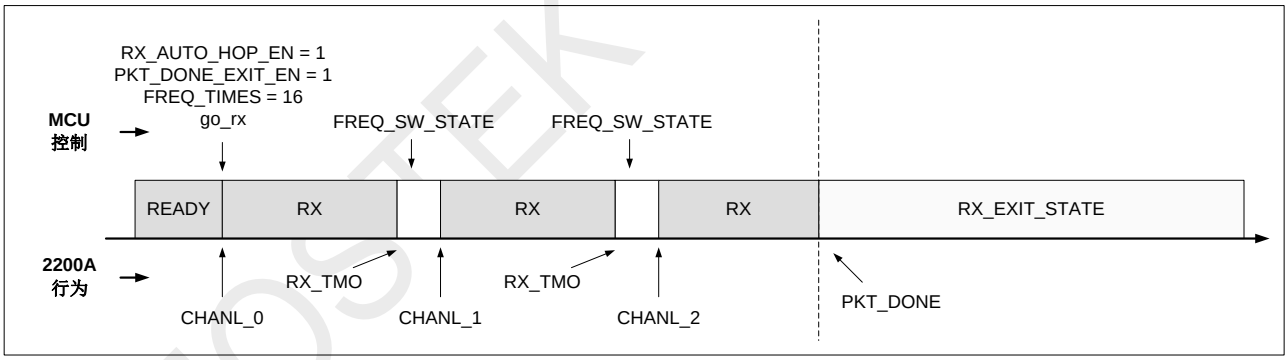
2. RX_HOP_PERSIST_EN = 1



3.3.2 RX Auto Hop 模式 0

当模式设为 0 的时候，接收时间等于 RX T1，不做任何延长处理。在 T1 超时前如果检测 PKT_DONE，并且 PKT_DONE_EXIT_EN = 1 即退出自动跳频，把状态切换到 RX_EXIT_STATE，此时 Rx Auto Hop 停止；否则 T1 超时退出 RX，返回 FREQ_SW_STATE 状态，再继续跳往下一个频点。

从下面的示意图中可见，MCU 只负责发送 go_rx 命令切换状态，CMT2310A 会进行自动进行频点切换和状态切换，并产生相应的中断。



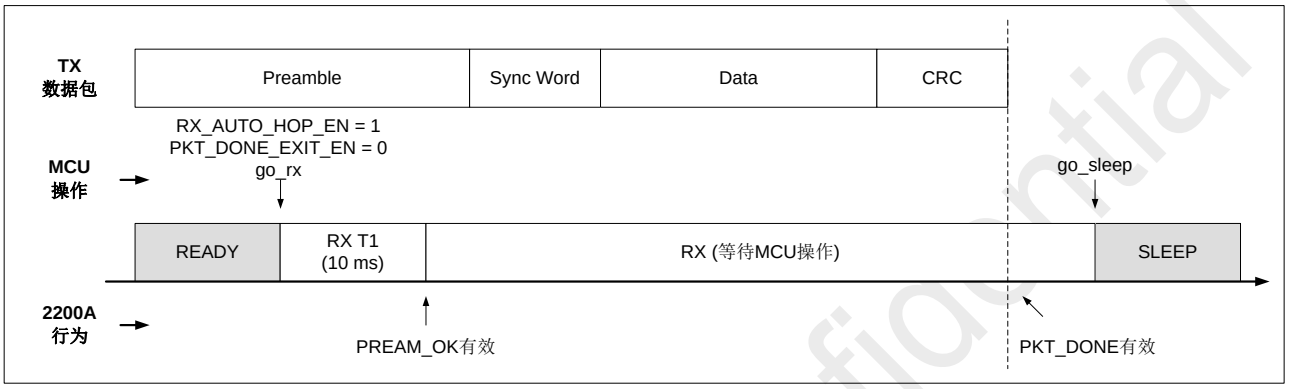
3.3.3 RX Auto Hop 模式 1-3

当模式设置为 1-3 的时候，RX T1 内一旦满足检测条件，RX T1 停止计时，芯片停留在 RX，并将控制权交给 MCU 或者自动切换到 RX_EXIT_STATE 定义的状态，此时 Rx Auto Hop 就停止了；否则 RX T1 计时结束后退出 RX 切换到 FREQ_SW_STATE 定义的状态，然后在继续跳往下一个频点。三种不同的条件如下：

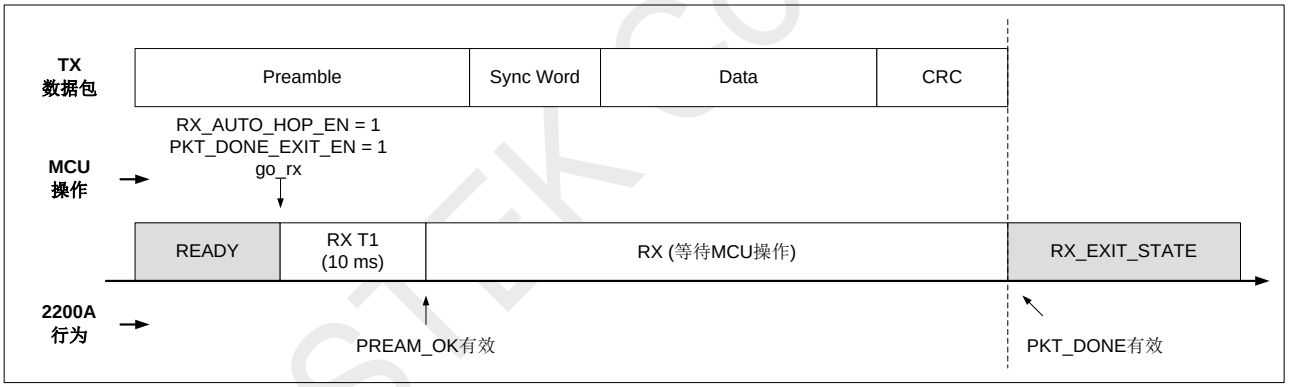
1：检测条件为 RSSI_PJD_VLD 有效
2：检测条件为 PREAM_OK 有效
3：检测条件为 RSSI_PJD_VLD 与 PREAM_OK 同时有效

下面以选项 2 为例，给出 TX，MCU 和 RX 配合，在一个频点中成功接收数据包的时序图。图 1 在成功收到数据包后，把控制权交给了 MCU。图 2 是收到数据包后自动切换到 RX_EXIT_STATE 定义的状态。

1. PKT_DONE_EXIT_EN = 0



2. PKT_DONE_EXIT_EN = 1



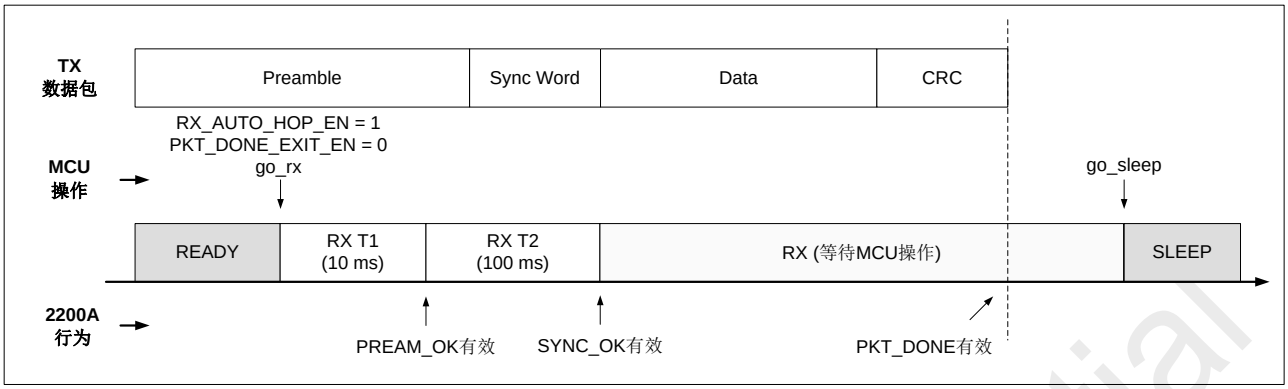
3.3.4 RX Auto Hop 模式 4-6

当模式设置为 4 - 6 的时候，RX T1 内一旦满足检测条件，即切换到 RX T2，RX T2 计时期间内检测到 SYNC_PASS 将控制权交给 MCU 或者切换到 RX_EXIT_STATE，此时 Rx Auto Hop 停止；否则 T2 计时结束后退出 RX，然后在继续跳往下一个频点。RX T1 之内检测不到条件满足，计时结束后也退出 RX，然后在继续跳往下一个频点。三种不同的条件如下：

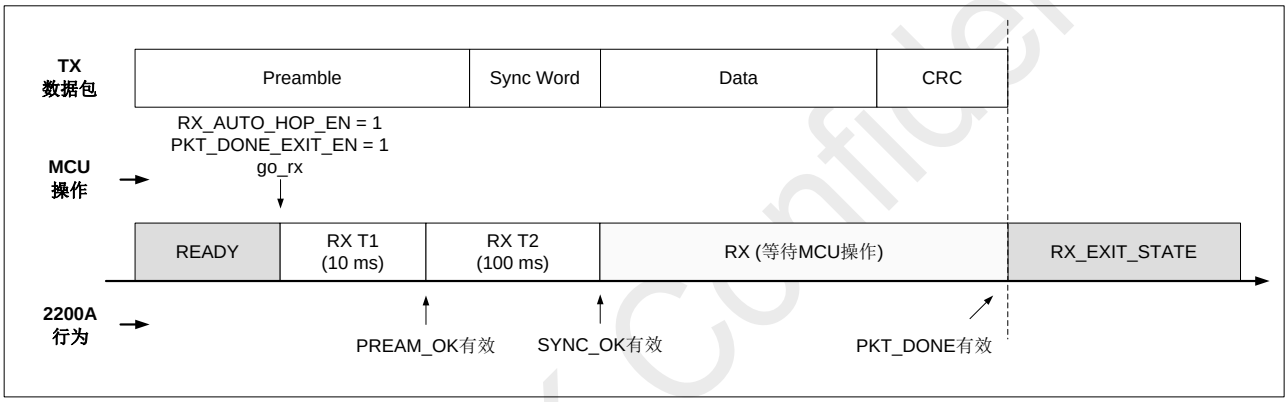
4：检测条件为 RSSI_PJD_VLD 有效
5：检测条件为 PREAM_OK 有效
6：检测条件为 RSSI_PJD_VLD 与 PREAM_OK 同时有效

下面以选项 5 为例，给出 TX，MCU 和 RX 配合工作的时序图：

1. PKT_DONE_EXIT_EN = 0



2. PKT_DONE_EXIT_EN = 1



4. 自动应答（ACK）

4.1 ACK 模式相关的寄存器

ACK 功能是实现发射机与接收机之间握手的一种自动控制机制，该功能提供两个包格式域用于判断发射端与接收端是否成功握手，分别是 SEQNUM 和 FCS2 域，这两个域属于 Packet 的一部分，使用该功能时需要通过 SEQNUM_EN 和 FCS2_EN 使能。

其实现原理为，发射端将本地的 SEQNUM 配置好，并将 FCS2 域中的 NACK 比特置 1 发射出去，发送成功后进入 RX 状态等待应答，当收到的应答包 SEQNUM 与本地 SEQNUM 匹配，并且收到的 NACK 比特为 1 时，认为接收方成功收到发射出去的包，认为握手成功。

接收端使能 ACK 功能后，进入 RX 状态，成功收到 packet 时通过收到的 NACK 比特判断是否需要应答，需要应答则进入 TX 状态，将收到的序列号和 NACK 比特回发，回发完成才产生 PKT_DONE 中断，接着根据 PKT_DONE_EXIT_EN 判断是回到 RX_EXIT_STATE 配置的状态，还是保持在 RX 状态。

4.2 TX ACK

相关寄存器及描述如下。

表 20. Tx ACK 相关寄存器

寄存器名	位数	R/W	比特名	功能说明
CTL_REG_7 (0x07)	4	RW	API_STOP	用户配置该比特用于停止 feature 功能，置 1 后并不是立刻生效，而是在芯片下一次切换状态时生效。feature 成功停止后返回到 Ready 状态。 0：不停止 feature 1：停止 feature
CTL_REG_23 (0x17)	7	RW	API_DONE_EN	api_done_flag 和 api_done 中断使能配置位，使能之后，api_done_flag 对应事件有效时，api_done_flag 会置 1，也会产生 api_done 中断。 0：不使能 1：使能
	3	RW	ACK_RECV_FAILED_EN	ack_recv_failed_flag 和 ack_recv_failed 中断使能配置位，使能之后，ack_recv_failed_flag 对应事件有效时，ack_recv_failed_flag 会置 1，也会产生 ack_recv_failed 中断。 0：不使能 1：使能
	2	RW	TX_RESEND_MAX_EN	tx_resend_max_flag 和 tx_resend_max 中断使能配置位，使能之后，tx_resend_max_flag 对应事件有效时，tx_resend_max_flag 会置 1，也会产生 tx_resend_max 中断。

寄存器名	位数	R/W	比特名	功能说明
				0：不使能 1：使能
	1	RW	NACK_RECV_EN	nack_recv_flag 和 nack_recv 中断使能配置位，使能之后，nack_recv_flag 对应事件有效时，nack_recv_flag 会置 1，也会产生 nack_recv 中断。 0：不使能 1：使能
	0	RW	SEQ_MATCH_EN	seq_match_flag 和 seq_match 中断使能配置位，使能之后，seq_match_flag 对应事件有效时，seq_match_flag 会置 1，也会产生 seq_match 中断。 0：不使能 1：使能
CTL_REG_31 (0x1F)	7	W	API_DONE_CLR	api_done_flag 清 0 位。 0：NA 1：将 api_done_flag 清 0
	3	W	ACK_RECV_FAILED_CLR	ack_recv_failed_flag 清 0 位。 0：NA 1：将 ack_recv_failed _flag 清 0
	2	W	TX_RESEND_MAX_CLR	tx_resend_max_flag 清 0 位。 0：NA 1：将 tx_resend_max_flag 清 0
	1	W	NACK_RECV_CLR	nack_recv_flag 清 0 位。 0：NA 1：将 nack_recv_flag 清 0
	0	W	SEQ_MATCH_CLR	seq_match_flag 清 0 位。 0：NA 1：将 seq_match_flag 清 0
CTL_REG_32 (0x20)	7	R	API_DONE_FLAG	api_done_flag 标志位，当 api_stop 比特被置 1 并被成功响应后，芯片返回到 READY 状态，芯片在 api_done_en 使能的情况下会将 api_done_flag 标志位置 1。 0：api_stop 未成功响应 1：api_stop 成功响应，停止 feature
	3	R	ACK_RECV_FAILED_FLAG	tx ack 模式下，成功收到应答包时，nack_recv_flag 不等于 1 或者 seq_match_flag 不等于 1，并且 ack_recv_failed_en 使能的情况下，会将 ack_recv_failed_flag 置 1。 0：tx ack 模式成功收到应答包 1：tx ack 模式未成功收到应答包
	2	R	TX_RESEND_MAX_FLAG	tx ack 模式带有 resend 功能，resend 次数用户可以自行配置，当重发次数达到配置的 resend 次数仍未收到应答，并且 tx_resend_max_en 使

寄存器名	位数	R/W	比特名	功能说明
				能的情况下，会将 tx_resend_max_flag 置 1。 0：tx ack 未达到最大重发次数 1：tx ack 达到最大重发次数
	1	R	NACK_RECV_FLAG	tx ack 模式下，收到应答包时并且 nack_recv_en 使能的情况下，会将收到 NACK 比特置于 nack_recv_flag。 0：收到应答包的 nack 比特为 0 1：收到应答包的 nack 比特为 1
	0	R	SEQ_MATCH_FLAG	tx ack 模式下，将 seq_match_en 使能后，收到应答包时，会将收到的 seq_num 与本地的 seq_num 作比较，匹配并且 seq_match_en 使能的情况下则将 seq_match_flag 置 1。 0：本地 seq_num 与收到的 seq_num 不匹配 1：本地 seq_num 与收到的 seq_num 匹配
CTL_REG_84 (0x54)	6	RW	FCS2_EN	配置 packet 是否带有 fcs2 域 0：packet 不带 fcs2 域 1：packet 带有 fcs2 域
	5	RW	SEQNUM_MATCH_EN	seqnum_en 使能并且 tx ack 模式使能，使能该比特会将本地 seq_num 与接收到的 seq_num 比较。 0：不使能 seq_num 比较 1：使能 seq_num 比较
	4	RW	SEQNUM_SIZE	配置 seq_num 的长度 0：1 byte 1：2 byte
	3	RW	SEQNUM_AUTO_INC	
	2	RW	SEQNUM_EN	配置 packet 是否带有 seqnum 域。 0：packet 不带 seqnum 域 1：packet 带有 seqnum 域
CTL_REG_87 (0x57)	7:0	RW	SEQNUM_TX_IN[7:0]	seqnum 低 8 位，tx ack 模式用户可自行配置 seqnum。
CTL_REG_88 (0x58)	7:0	RW	SEQNUM_TX_IN[15:8]	seqnum 高 8 位，tx ack 模式用户可自行配置 seqnum。
CTL_REG_89 (0x59)	7:0	RW	SEQNUM_TX_OUT[7:0]	实际 tx packet 的 seqnum 的低 8 位。
CTL_REG_90 (0x5A)	7:0	RW	SEQNUM_TX_OUT[15:8]	实际 tx packet 的 seqnum 的高 8 位。
CTL_REG_91 (0x5B)	7:0	RW	FCS2_TX_IN[7:0]	用户可自行配置 fcs2 域的值。
CTL_REG_92 (0x5C)	7:0	RW	FCS2_RX_OUT[7:0]	收到 packet 的 fcs2 域的值。
CTL_REG_96	6:4	RW	TX_EXIT_STATE<2:0>	完成发射后自动退出到设定的状态，只在 packet 模式下有效，否则芯片不会自动退出 TX 状态，

寄存器名	位数	R/W	比特名	功能说明
(0x60)				而是等待 MCU 发 go_*命令来切换。 1：SLEEP 2：READY 3：TFS 4：TX 5：RFS 6：RX Others：SLEEP
	2	RW	TX_ACK_EN	tx ack 功能使能位。 1：使能 0：不使能
CTL_REG_97 (0x61)	6:4	RW	RX_TIMEOUT_STATE[2:0]	rxtimer timeout 之后返回的状态。 1：SLEEP 2：READY 3：TFS 4：TX 5：RFS 6：RX Others：SLEEP
	3	RW	TIMER_RX_EN	RX TIMER 的使能。 0：不使能 1：使能
CTL_REG_98 (0x62)	7	RW	PKT_DONE_EXIT_EN	芯片成功收到 packet 产生 pkt_done 时是保持当前状态还是立即退出返回到 RX_EXIT_STATE 对应的状态。 0：芯片保持当前状态 1：芯片根据 RX_EXIT_STATE 配置返回到对应状态
	3:0	RW	SLP_MODE<3:0>	RX Duty Cycle 模式配置。 有效范围 0-13
CTL_REG_101 (0x65)	7:0	RW	RX_TIMER_T1_M<7:0>	定义了 RX T1 TIMER 的计时时间，公式如下： T = M x 2^(R+1) x 20 us R 的取值范围是 0-21。
CTL_REG_102 (0x66)	7:5	RW	RX_TIMER_T1_M<10:8>	
	4:0	RW	RX_TIMER_T1_R<4:0>	
CTL_REG_113 (0x71)	7:0	RW	TX_RS_TIMES<7:0>	TX ACK 模式下，规定的最大重发次数。
CTL_REG_114 (0x72)	7:0	R	TX_RS_DONE_TIMES<7:0>	TX ACK 模式下已经重发的次数。

4.2.1 TX ACK 功能使用说明

TX ACK 模式提供四个中断标志位供用户查询芯片当前状况。分别是 ACK_RECV_FAILED_FLAG、TX_RESEND_MAX_FLAG、NACK_RECV_FLAG 和 SEQ_MATCH_FLAG。ACK_RECV_FAILED_EN、TX_RESEND_MAX_EN、NACK_RECV_EN 和 SEQ_MATCH_EN 置 1 的情况下，相应事件发生时，对应的中断标志位会置 1。

ACK_RECV_FAILED_FLAG 会在芯片收到应答包但是比较信息不匹配时置 1。TX_RESEND_MAX_FLAG 会在芯片执行 TX_RS_TIMES 配置的重发次数后仍未收到应答包时置 1。NACK_RECV_FLAG 的值由收到应答包 FCS2 域中的 NACK 位决定。SEQ_MATCH_FLAG 会在 SEQ_MATCH_EN 使能并收到应答包 SEQNUM 与本地的 SEQNUM 一致时置 1。

ACK 模式提供两个包格式域用于判断发射端与接收端是否成功握手，分别是 SEQNUM 和 FCS2 域，这两个域属于 packet 的一部分，分别通过 SEQNUM_EN 和 FCS2_EN 使能。实现的原理是发射端将本地的 SEQNUM 配置好，并将 FCS2 域中的 NACK 比特置 1（表示需要应答）发射出去，发送成功后进入 RX 状态等待应答，当收到的应答包 SEQNUM 与本地 SEQNUM 匹配，并且收到的 NACK 比特为 1 时，认为接收方成功收到发射出去的包，认为握手成功。

SEQNUM 是否需要匹配由 SEQNUM_MATCH_EN 控制，使能后在收到应答包时会做匹配，匹配成功 SEQ_MATCH_FLAG 会置 1；不使能不做匹配。SEQNUM 的大小可通过 SEQNUM_SIZE 配置，范围是 1-2 字节。SEQNUM 的值可以通过配置 SEQNUM_TX_IN<15:0>来进行初始化。SEQNUM 有自加 1 的功能，通过 SEQNUM_AUTO_INC 使能，使能之后，每次发射前会将 SEQNUM 在原值的基础上加 1，然后再发送出去。SEQNUM_TX_OUT<15:0>可供用户查询当前发送的 SEQNUM 的值。

NACK 比特在 FCS2 域的最高位，其他位为保留位，FCS2 域可以通过 FCS2_TX_IN<7:0>配置，FCS2 域使能的情况下，收到包时会将收到的 FCS2 域的内容放到 FCS2_RX_OUT<7:0>寄存器中。

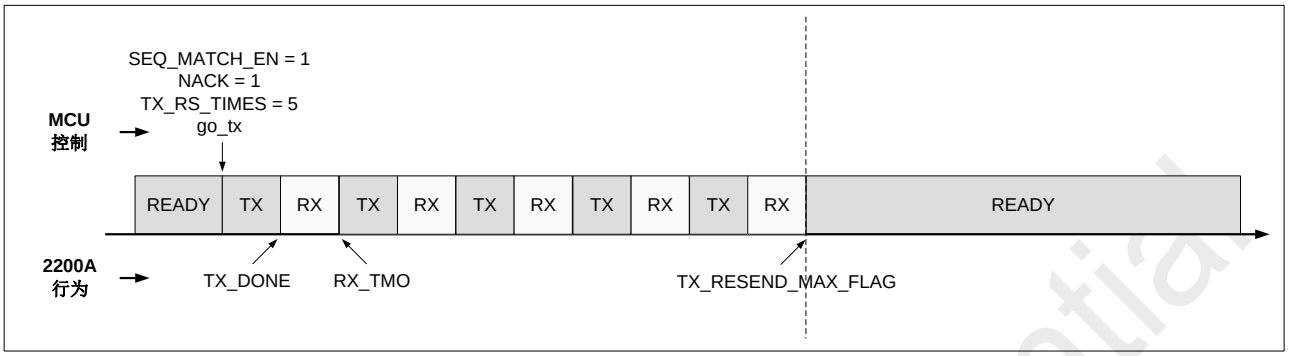
TX ACK 模式自动过程涉及到 RX T1 TIMER，所以使用过程中必须将 TIMER_RX_EN 置 1。RX T1 TIMER 通过 TIMER_M_RX_T1 和 TIMER_R_RX_T1 配置，实际定时时间可以由公式计算得出。PKT_DONE_EXIT_EN 决定在收到 PKT_DONE 时是否要退出返回到 RX_EXIT_STATE 配置的状态，还是保持在当前状态。

TX ACK 模式提供重发机制，当没有成功收到应答包时，可以重新进入 TX 然后等待应答。通过 TX_RESEND_TIMES<7:0>配置重发次数，当达到最大重发次数仍未收到应答包时，TX_RESEND_MAX_FLAG 标志位置 1。TX_RESEND_DONE_TIMES<7:0>表示已经完成的重发次数。

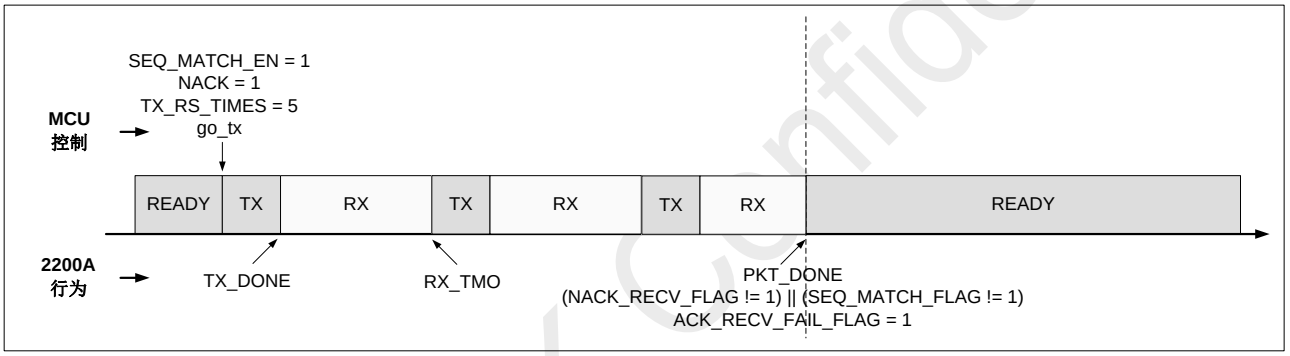
用户完成配置之后，将 TX_ACK_EN 置 1 使能，使能之后，发送 go_tx 命令就可以进入了 TX ACK 模式，自动运行过程中如果触发停止事件，芯片会退出自动过程停止到固定的某个状态，如果用户想自行停止 TX ACK 功能，可以将 API_STOP 置 1，芯片检测到该比特之后，会在下一个状态切换时将 API_DONE_FLAG 置 1，保持当前配置然后退出 TX ACK 返回到 READY 状态。

4.2.2 TX ACK 时序图说明

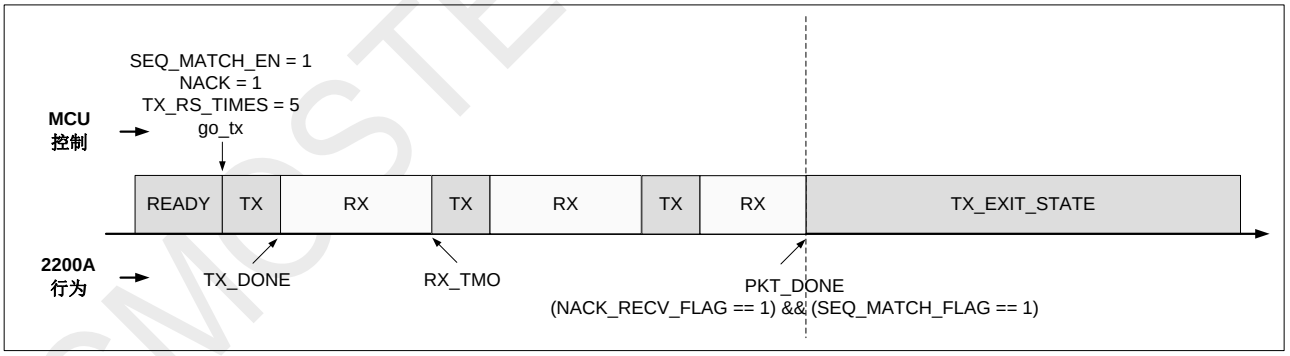
1. 达到配置的最大重发次数仍未成功收到应答



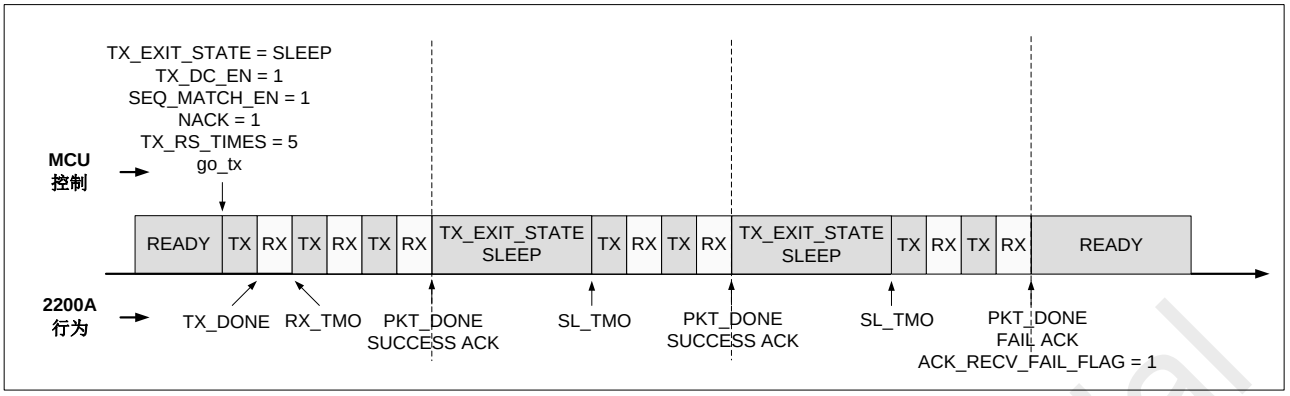
2. 成功接收到应答包但是内容不匹配



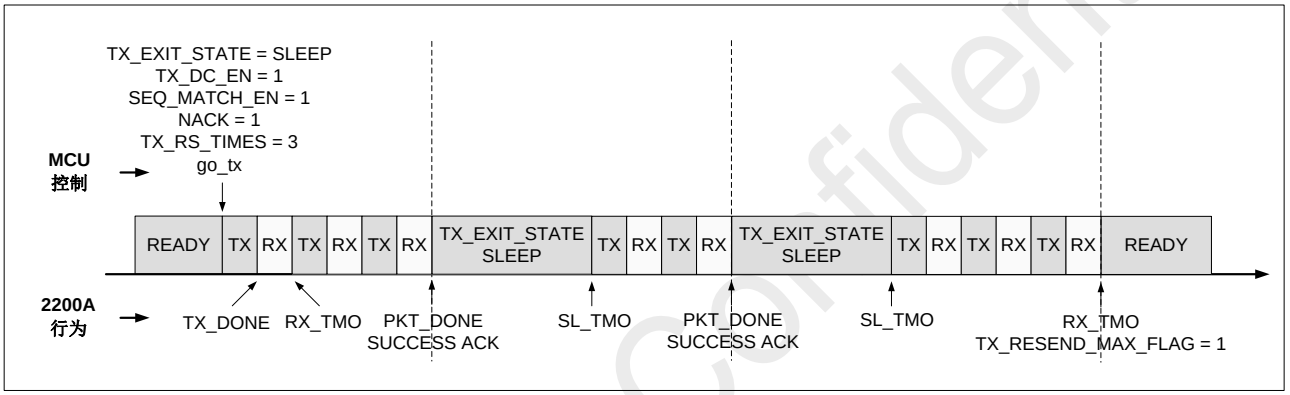
3. 成功收到应答包并且内容匹配



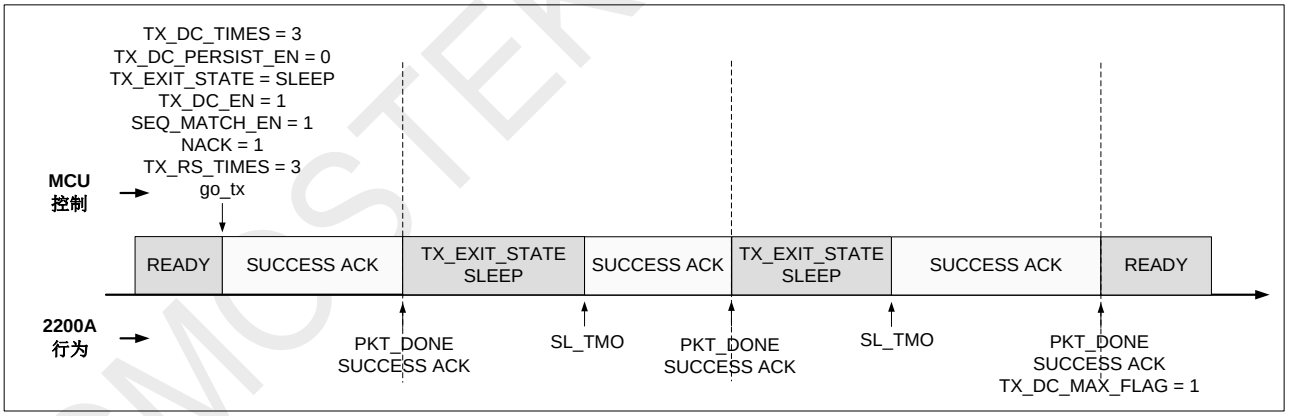
4. TX_ACK 与 TX_DUTY_CYCLE 结合，等待应答过程中收到错误应答包，失败退出



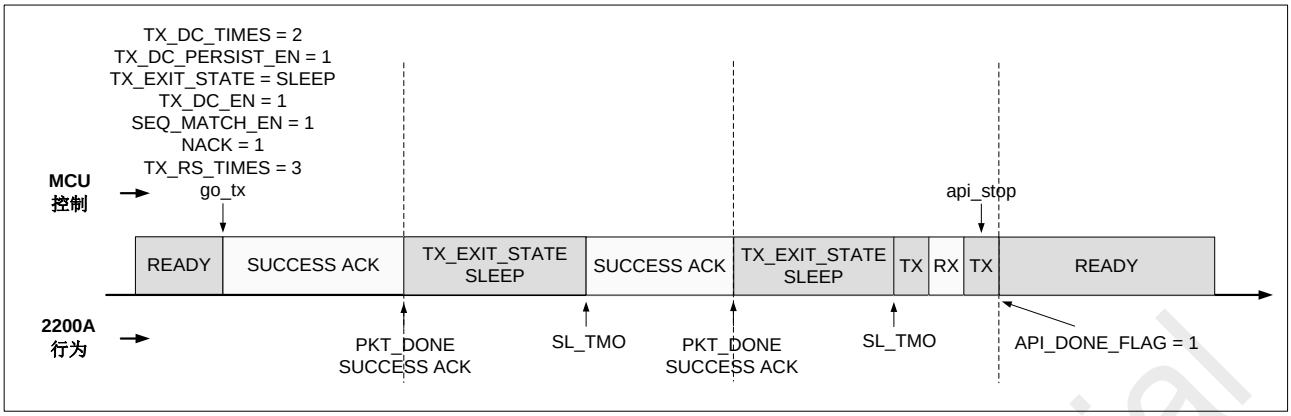
5. TX_ACK 与 TX_DUTY_CYCLE 结合，重发过程中达到最大次数，失败退出



6. TX_ACK 与 TX_DUTY_CYCLE 结合，过程中应答都成功，达到 TX_DC_TIMES 后退出



7. TX_ACK 与 TX_DUTY_CYCLE 结合，TX DC PERSISTENT 模式下，通过 api_stop 退出



4.3 RX ACK

相关寄存器及描述如下。

表 21. RX ACK 相关寄存器

寄存器名	位数	R/W	比特名	功能说明
CTL_REG_7 (0x07)	4	RW	API_STOP	用户配置该比特用于停止 feature 功能，置 1 后并不是立刻生效，而是在芯片下一次切换状态时生效。feature 成功停止后返回到 Ready 状态。 0：不停止 feature 1：停止 feature
CTL_REG_23 (0x17)	7	RW	API_DONE_EN	api_done_flag 和 api_done 中断使能配置位，使能之后，api_done_flag 对应事件有效时，api_done_flag 会置 1，也会产生 api_done 中断。 0：不使能 1：使能
CTL_REG_31 (0x1F)	7	RW	API_DONE_CLR	api_done_flag 清 0 位 0：NA 1：将 api_done_flag 清 0
CTL_REG_32 (0x20)	7	R	API_DONE_FLAG	api_done_flag 标志位，当 api_stop 比特被置 1 并被成功响应后，芯片返回到 READY 状态，芯片在 api_done_en 使能的情况下会将 api_done_flag 标志位置 1。 0：api_stop 未成功响应 1：api_stop 成功响应，停止 feature
CTL_REG_38 (0x26)	7:0	R	RX_SEQNUM<7:0>	RX 模式时接收到的包序列号低 8 位。
CTL_REG_39 (0x27)	7:0	R	RX_SEQNUM<15:8>	RX 模式时接收到的包序列号高 8 位。
CTL_REG_84 (0x54)	6	RW	FCS2_EN	配置 packet 是否带有 fcs2 域。 1：packet 带有 fcs2 域

寄存器名	位数	R/W	比特名	功能说明
				0：packet 不带 fcs2 域
	5	RW	SEQNUM_MATCH_EN	seqnum_en 使能并且 tx ack 模式使能，使能该比特会将本地 seq_num 与接收到的 seq_num 比较。 1：使能 seq_num 比较 0：不使能 seq_num 比较
	4	RW	SEQNUM_SIZE	配置 seq_num 的长度 1：2 byte 0：1 byte
	3	RW	SEQNUM_AUTO_INC	
	2	RW	SEQNUM_EN	配置 packet 是否带有 seqnum 域。 1：packet 带有 seqnum 域 0：packet 不带 seqnum 域
CTL_REG_87 (0x57)	7:0	RW	SEQNUM_TX_IN[7:0]	seqnum 低 8 位，tx ack 模式用户可自行配置 seqnum。
CTL_REG_88 (0x58)	7:0	RW	SEQNUM_TX_IN[15:8]	seqnum 高 8 位，tx ack 模式用户可自行配置 seqnum。
CTL_REG_89 (0x59)	7:0	RW	SEQNUM_TX_OUT[7:0]	实际 tx packet 的 seqnum 的低 8 位。
CTL_REG_90 (0x5A)	7:0	RW	SEQNUM_TX_OUT[15:8]	实际 tx packet 的 seqnum 的高 8 位。
CTL_REG_91 (0x5B)	7:0	RW	FCS2_TX_IN[7:0]	用户可自行配置 fcs2 域的值。
CTL_REG_92 (0x5C)	7:0	RW	FCS2_RX_OUT[7:0]	收到 packet 的 fcs2 域的值。
CTL_REG_97 (0x61)	7	RW	CSMA_EN	csma 功能使能位。 1：使能 0：不使能
	6:4	RW	RX_TIMEOUT_STATE[3:0]	Rx timer timeout 之后返回的状态。 有效范围 0-13。
	3	RW	TIMER_RX_EN	rx timer 使能位，使能后 rx timer 根据配置开始工作。 1：使能 0：不使能
	2	RW	RX_ACK_EN	rx ack 功能使能位。 1：使能 0：不使能
CTL_REG_98 (0x62)	7	RW	PKT_DONE_EXIT_EN	芯片成功收到 packet 产生 pkt_done 时是保持当前状态还是立即退出返回到 RX_EXIT_STATE 对应的状态。 0：芯片保持当前状态 1：芯片根据 RX_EXIT_STATE 配置返回到对应状态

寄存器名	位数	R/W	比特名	功能说明
	2:0	RW	DUTY_CYCLE_METHOD[3:0]	Rx Duty Cycle 模式配置。 有效范围 0-13。
CTL_REG_101 (0x65)	7:0	RW	RX_TIMER_T1_M<7:0>	定义了 RX T1 TIMER 的计时时间，公式如下： T = M x 2^(R+1) x 20 us R 的取值范围是 0-21。
CTL_REG_102 (0x66)	7:5	RW	RX_TIMER_T1_M<10:8>	
	4:0	RW	RX_TIMER_T1_R<4:0>	
CTL_REG_103 (0x67)	7:0	RW	RX_TIMER_T2_M<7:0>	定义了 RX T1 TIMER 的计时时间，公式如下： T = M x 2^(R+1) x 20 us R 的取值范围是 0-21。
CTL_REG_104 (0x68)	7:5	RW	RX_TIMER_T2_M<10:8>	
	4:0	RW	RX_TIMER_T2_R<4:0>	

4.3.1 RX ACK 功能使用说明

接收端使能并收到包之后，首先判断 FCS2 域中的 NACK 比特是否为 1，为 1 的话切换回 TX，将收到的 SEQNUM 和自身配置的 FCS2 域回发给发送端。回发可以是带 SEQNUM 和 FCS2 域而没有 payload 内容的简包，或者是带 SEQNUM 和 FCS2 域并且有 payload 内容的搭车包。需要注意的是，搭车包情况下，发送和接收端都应该配置为可变长度模式。简包还是搭车包由 PIGGYBACKING 配置，PIGGYBACKING 为 1 时是搭车包，payload 内容来自于 TX FIFO，所以需要在回发前填好 TX FIFO 的内容。

SEQNUM 的大小可通过 SEQNUM_SIZE 配置，范围是 1-2 字节。SEQNUM 可以由用户自行初始化，通过 SEQNUM_TX_IN<15:0>配置。SEQNUM 有自加 1 的功能，通过 SEQNUM_AUTO_INC 使能，使能之后，每次发射前会将 SEQNUM 在原值的基础上加 1，然后再发送出去。SEQNUM_TX_OUT<15:0>可供用户查询当前发送的 SEQNUM 的值。接收到的 SEQNUM 会存入 RX_SEQNUM<15:0>供用户读取。

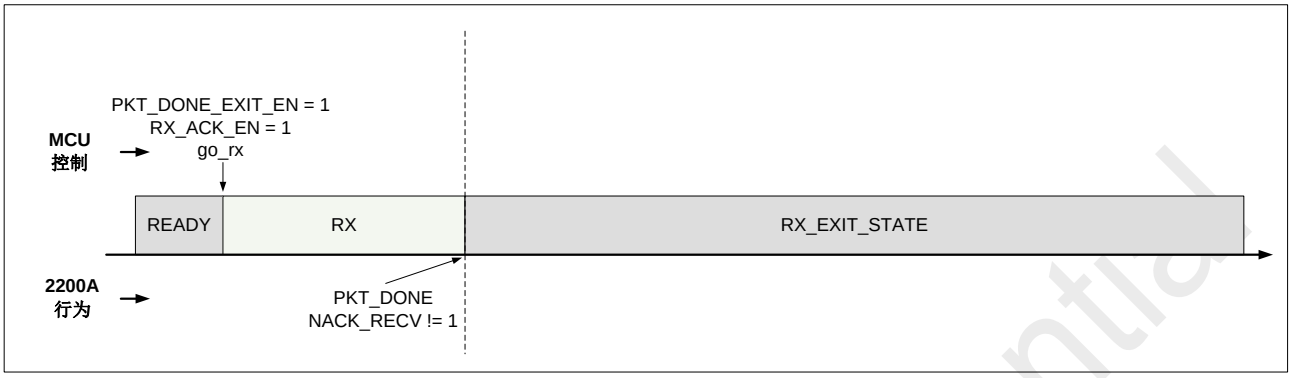
NACK 比特在 FCS2 域的最高位，其他位为保留位，FCS2 域可以通过 fcs2_tx_in<7:0>配置，FCS2 域使能的情况下，收到包时会将收到的 FCS2 域内容放到 FCS2_RX_OUT<7:0>中。

RX ACK 模式自动过程涉及到 RX T1 TIMER，所以使用过程中必须将 TIMER_RX_EN 置 1。RX T1 TIMER 通过 TIMER_M_RX_T1 和 TIMER_R_RX_T1 配置，实际定时时间可以由公式计算得出。在需要回发的情况下，回发完成之后才认为接收机 PKT_DONE，PKT_DONE_EXIT_EN 决定在收到 PKT_DONE 时是否要退出返回到 RX_EXIT_STATE 配置的状态，还是保持在 RX 状态。

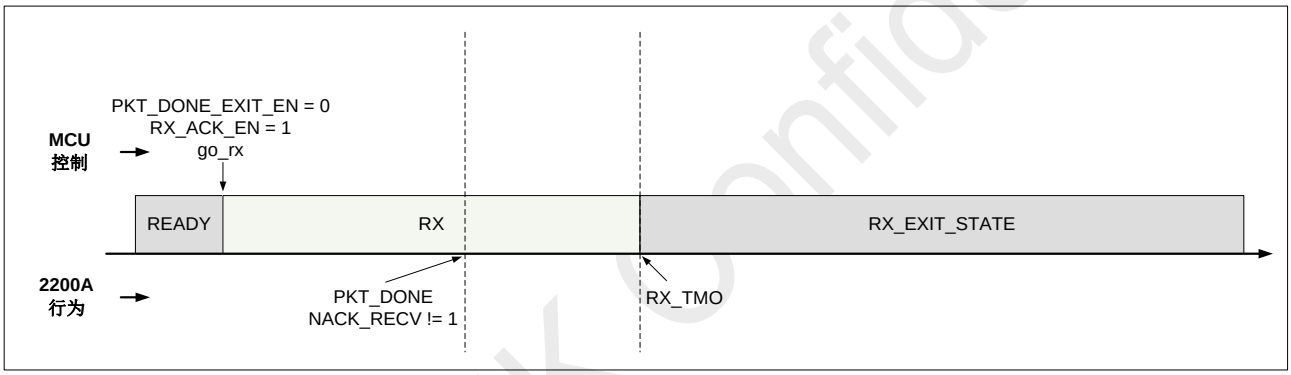
用户完成配置之后，将 RX_ACK_EN 置 1 使能，使能之后，发送 go_rx 命令就可以进入了 RX ACK 模式，自动运行过程中如果触发停止事件，芯片会退出自动过程停止到固定的某个状态，如果用户想自行停止 RX ACK 功能，可以将 API_STOP 置 1，芯片检测到该比特之后，会在下一个状态切换时将 API_DONE_FLAG 置 1，保持当前配置然后退出 RX ACK 返回到 READY 状态。

4.3.2 RX ACK 时序图说明

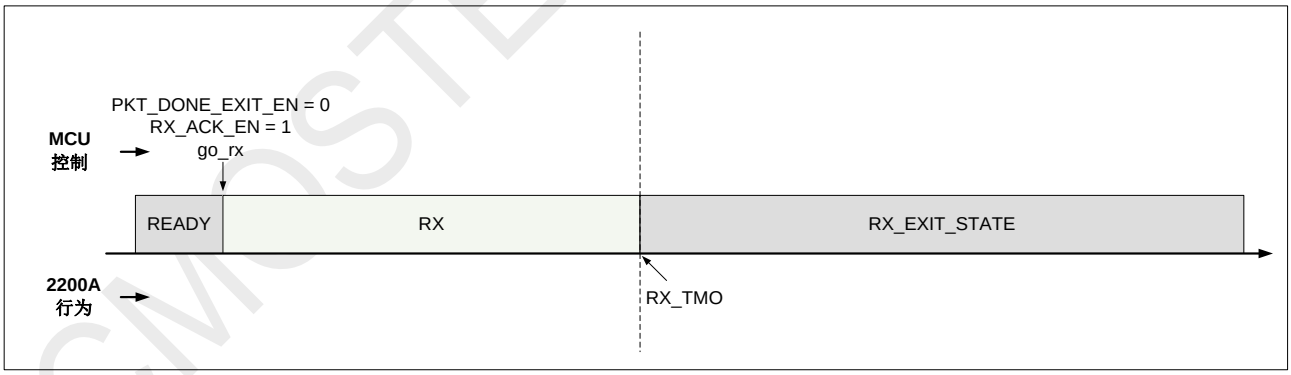
1. RX_ACK 收到包，但是应答 bit 为 0，同时 PKT_DONE_EXIT_EN = 1



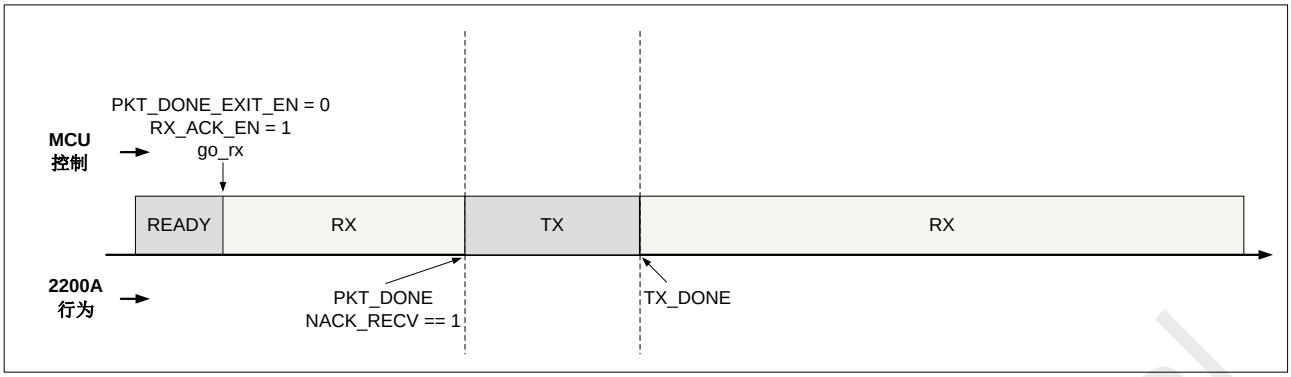
2. RX_ACK 收到包，但是应答 bit 为 0，同时 PKT_DONE_EXIT_EN = 0



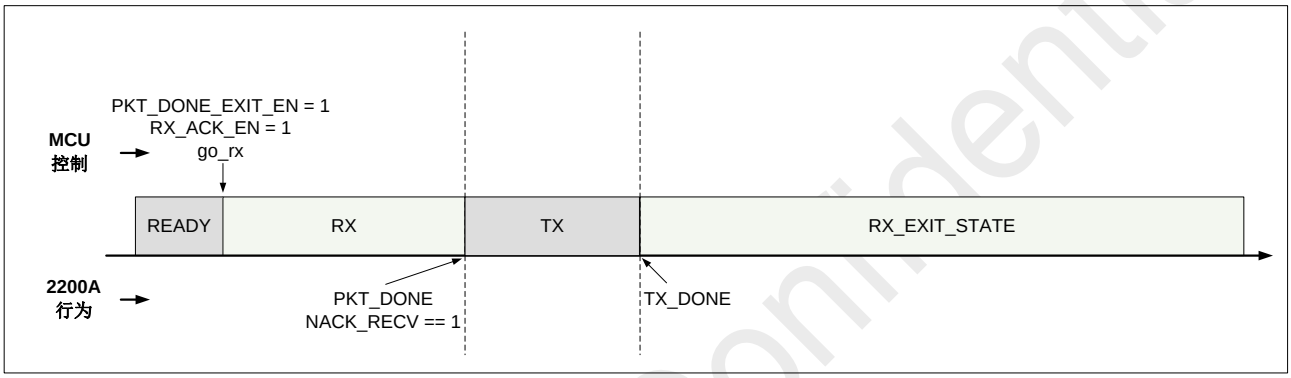
3. RX_ACK 没收到包，RX TIMEOUT 退出



4. RX_ACK 收到包，应答比特为 1，同时 PKT_DONE_EXIT_EN = 0



5. RX_ACK 收到包，应答比特为 1，同时 PKT_DONE_EXIT_EN = 1



5. CSMA

5.1 CSMA 模式相关的寄存器

寄存器的内容及描述如下。

表 22. 位于配置区的寄存器

寄存器名	位数	R/W	比特名	功能说明
CTL_REG_7 (0x07)	4	RW	API_STOP	用户配置该比特用于停止 feature 功能，置 1 后并不是立刻生效，而是在芯片下一次切换状态时生效。feature 成功停止后返回到 Ready 状态。 0：不停止 feature 1：停止 feature
CTL_REG_23 (0x17)	7	RW	API_DONE_EN	api_done_flag 和 api_done 中断使能配置位，使能之后，api_done_flag 对应事件有效时，api_done_flag 会置 1，也会产生 api_done 中断。 0：不使能 1：使能
	6	RW	CCA_STATUS_EN	cca_status_flag 和 cca_status 中断使能配置位，使能之后，cca_status_flag 对应事件有效时，cca_status_flag 会置 1，也会产生 cca_status 中断。 0：不使能 1：使能
	5	RW	CSMA_MAX_EN	csma_max_flag 和 csma_max 中断使能配置位，使能之后，csma_max_flag 对应事件有效时，csma_max_flag 会置 1，也会产生 csma_max 中断。 0：不使能 1：使能
CTL_REG_31 (0x1F)	7	W	API_DONE_CLR	api_done_flag 清 0 位。 0：NA 1：将 api_done_flag 清 0
	6	W	CCA_STATUS_CLR	cca_status_flag 清 0 位。 0：NA 1：将 cca_status_flag 清 0
	5	W	CSMA_MAX_CLR	csma_max_flag 清 0 位。 0：NA 1：将 csma_max_flag 清 0
CTL_REG_32 (0x20)	7	R	API_DONE_FLAG	api_done_flag 标志位，当 api_stop 比特被置 1 并被成功响应后，芯片返回到 READY 状态，芯片在 api_done_en 使能的情况下会将 api_done_flag 标志位置 1。

寄存器名	位数	R/W	比特名	功能说明
				0：api_stop 未成功响应 1：api_stop 成功响应，停止 feature
	6	R	CCA_STATUS_FLAG	cca_status_flag 标志位，当使用 csma 功能时，首先进入 RX 状态检测当前信道是否空闲，如果当前信道忙并且 cca_status_en 已经使能，那么会将 cca_status_flag 置 1。 0：当前信道空闲 1：当前信道忙
	5	R	CSMA_MAX_FLAG	csma 功能可以配置信道最大信道侦听次数，如果达到最大侦听次数信道仍非空闲并且 csma_max_en 使能，那么会将 csma_max_flag 标志位置 1 。 0：未达到最大信号侦听次数 1：达到最大信道侦听次数
CTL_REG_96 (0x60)	6:4	RW	TX_EXIT_STATE[2:0]	tx 完成之后返回的状态。 1：SLEEP 2：READY 3：TFS 4：TX 5：RFS 6：RX Others：SLEEP
CTL_REG_97 (0x61)	7	RW	CSMA_EN	csma 功能使能位。 0：不使能 1：使能
	6:4	RW	RX_EXIT_STATE[2:0]	Rx 完成之后返回的状态。 1：SLEEP 2：READY 3：TFS 4：TX 5：RFS 6：RX Others：SLEEP
	3	RW	TIMER_RX_EN	rx timer 使能位 。 0：不使能 1：使能
CTL_REG_98 (0x62)	3:0	RW	DUTY_CYCLE_METHOD[3:0]	rx duty cycle 模式配置。 有效范围 0-13
CTL_REG_105 (0x69)	5:4	RW	TIMER_RANDOM_MODE[1:0]	配置 csma 模式下，csma sleep timer 的模式。 00：随机 R 值 01：随机 M 值 10：R 值和 M 值都随机 11：使用客户配置的固定值
	3	RW	TIMER_SLEEP_EN	sleep timer 使能位。

寄存器名	位数	R/W	比特名	功能说明
				0：不使能 1：使能
CTL_REG_105 (0x6A)	7:5	RW	CCA_MODE[2:0]	cca_status 产生配置。 000：认为信道一直空闲。 001：4 个检测窗口有大于等于 1 次检测到 RSSI_OK，认为信道忙。 010：4 个检测窗口有大于等于 1 次检测到 PJD_OK，认为信道忙。 011：4 个检测窗口有大于等于 1 次检测到 RSSI_OK 或 PJD_OK，认为信道忙。 100：检测到 1 次 SYNC_PASS，认为信道忙。 101：检测到 1 次 SYNC_PASS ，或 4 个检测窗口有大于等于 1 次检测到 RSSI_OK，认为信道忙。 110：检测到 1 次 SYNC_PASS ，或 4 个检测窗口有大于等于 1 次检测到 PJD_OK，认为信道忙。 111：检测到 1 次 SYNC_PASS ，或 4 个检测窗口有大于等于 1 次检测到 RSSI_OK 或 PJD_OK，认为信道忙。
	4	RW	CSMA_PERSIST_EN	该比特置 1 后，芯片不管 csma_times 配置，一直检测信道空闲情况直到成功发射数据包出去。 1：持续检测直至信道空闲并发射成功 0：csma 不持续检测
	1:0	RW	CCA_WIN_SEL[1:0]	csma 每个检测窗口大小。 00：32 symbol 01：64 symbol 10：128 symbol 11：256 symbol
CTL_REG_107 (0x6B)	7:0	RW	TIMER_M_RX_CSMA [7:0]	定义了 rx csma timer 的计时时间，公式如下： T = M x 2^(R+1) x 5us
CTL_REG_108 (0x6C)	7:5	RW	TIMER_M_RX_CSMA [10:8]	
	4:0	RW	TIMER_R_RX_CSMA [4:0]	
CTL_REG_115 (0x73)	7:0	RW	CSMA_TIMES[7:0]	配置 csma 信道最大检测次数，范围 0-255。
CTL_REG_116 (0x74)	7:0	RW	CSMA_DONE_TIMES[7:0]	csma 已经完成检测的次数，范围 0-255。
CTL_REG_118 (0x76)	7:0	RW	TIMER_M_SLEEP_CSMA[7:0]	定义了 sleep csma timer 的计时时间，公式如下： T = M x 2^(R+1) x 31.25us
CTL_REG_119 (0x77)	7:5	RW	TIMER_M_SLEEP_CSMA[10:8]	
	4:0	RW	TIMER_R_SLEEP_CSMA[4:0]	

5.2 功能使用说明

CSMA 是一种在传输发射前预先感知当前信道闲忙情况再决定是否发送的一种冲突避免机制（先听后说），这避免了不同发射机同时使用相同的信道，并增加了接收机正确接收到数据包的概率。该机制的核心是通过将 RSSI 与设定的阈值做比较，从而判断信道闲忙情况。信道忙时，回退（back-off）到侦听的过程会被重复执行一定次数，直到信道被发现是空闲的。如果达到限制的次数都不成功，则产生对应的中断和标志位，停止发送。回退（back-off）过程中，芯片处于 SLEEP 状态。当然，CSMA 有 PERSISTENT 模式，PERSISTENT 模式下不是回退到 SLEEP 状态，而是回退到 RFS 后再切回 RX 状态侦听。该模式下会一直检测信道直到信道空闲把数据发送出去。非 PERSISTENT 模式下，可以通过 CSMA_TIMES 配置在信道非空闲情况下最大侦听的次数。如果达到最大侦听次数信道仍繁忙，则退出 CSMA 返回到 READY 状态。

CSMA 模式提供两个中断标志位供用户查询芯片当前状况。分别是 CCA_STATUS_FLAG 和 CSMA_MAX_FLAG。CCA_STATUS_EN 和 CSMA_MAX_EN 置 1 的情况下，相应事件发生时，CCA_STATUS_FLAG 和 CSMA_MAX_FLAG 会置 1。CCA_STATUS_FLAG 会在芯片进入 RX 侦听当前信道发现当前信道繁忙时置 1，CSMA_MAX_FLAG 会在芯片执行 CSMA_TIMES 配置的侦听次数后发现信道仍处于繁忙状态时置 1。

CSMA 模式自动过程涉及到 RX TIMER 和 SLEEP TIMER，所以使用过程中必须将 TIMER_RX_EN 和 TIMRE_SLEEP_EN 置 1。另外，CSMA 模式提供独立的 RX TIMER 和 SLEEP TIMER 定时配置，RX TIMER 通过 TIMER_M_RX_CSMA 和 TIMER_M_RX_CSMA 配置，实际定时时间可以由公式计算得出；SLEEP TIMER.通过 TIMER_M_SLEEP_CSMA 和 TIMER_M_SLEEP_CSMA 配置，实际定时时间可以由公式计算得出。

CSMA 分为 PERSISTENT 模式和非 PERSISTENT 模式，通过 CSMA_PERSIST_EN 配置。非 PERSISTENT 模式下，CSMA_TIMES 是允许的最大侦听次数，CSMA_DONE_TIMES 是已经完成的侦听次数，每次侦听结束 CSMA_DONE_TIMES 会自动加 1，当达到最大侦听次数 CSMA_TIMES 仍发现信道忙，那么 CSMA_MAX_FLAG 会置 1，然后退出 CSMA 模式。PERSISTENT 模式下，无论 CSMA_TIMES 和 CSMA_DONE_TIMES 关系，芯片会一直侦听信道，直到信道空闲将数据发送出去。

CSMA 模式下信道繁忙的评判标准由 CCA_MODE 决定，具体见 CTL_REG_105（0x6A）的寄存器描述。CSMA 模式下，信道的侦听有多个检测窗口，每个窗口大小由 CCA_WIN_SEL 决定，具体见 CTL_REG_105（0x6A）寄存器描述。

CSMA 模式下 SLEEP TIMER 定时时间由 TIMER_M_SLEEP_CSMA 和 TIMER_M_SLEEP_CSMA 决定，但它并不是固定的，通过 TIMER_RANDOM_MODE 可以配置 SLEEP TIMER 为随机时间，具体见 0CTL_REG_105（0x69）寄存器说明。

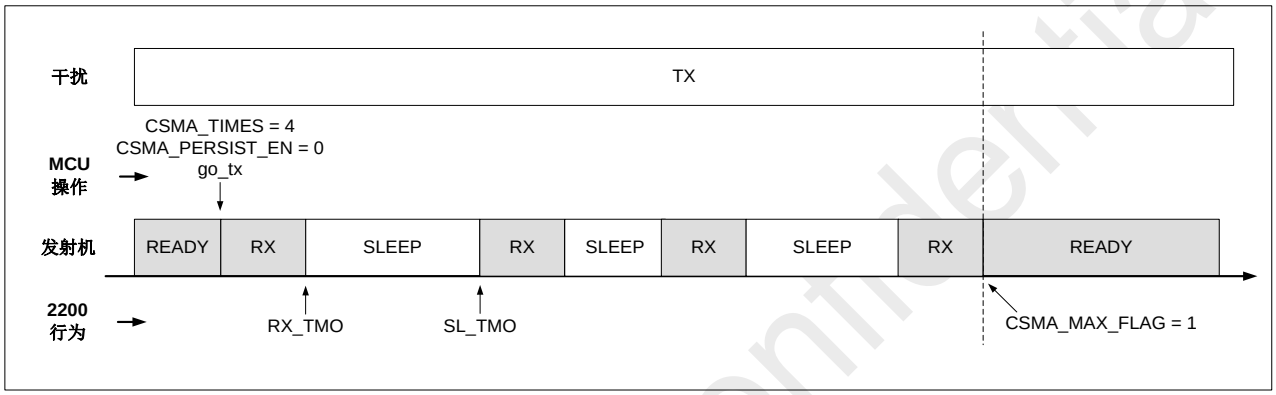
用户完成配置之后，将 CSMA_EN 置 1 使能，使能之后，发送 go_tx 命令就可以进入了 CSMA 模式，自动运行过程中如果触发停止事件，芯片会退出自动过程停止到固定的某个状态，如果用户想立即停止 CSMA 功能，可以将 API_STOP 置 1，芯片检测到该比特之后，会在下一个状态切换时将 API_DONE_FLAG 置 1，保持当前配置然后退出 CSMA 返回到 READY 状态。

5.3 时序图说明

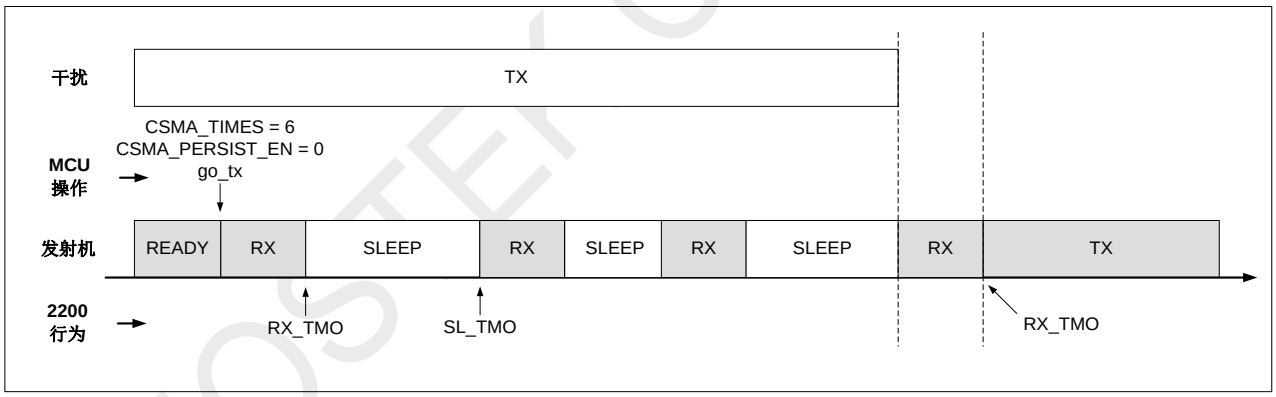
5.3.1 CSMA 侦听方式

CSMA 有两种侦听处理方式：当 CSMA_PERSIST_EN 为 0 时，芯片达到配置的最大侦听次数信道仍繁忙的话返回到 READY 状态；当 CSMA_PERSIST_EN 为 1 时，芯片一直检测当前信道直到信道空闲将数据发送出去。

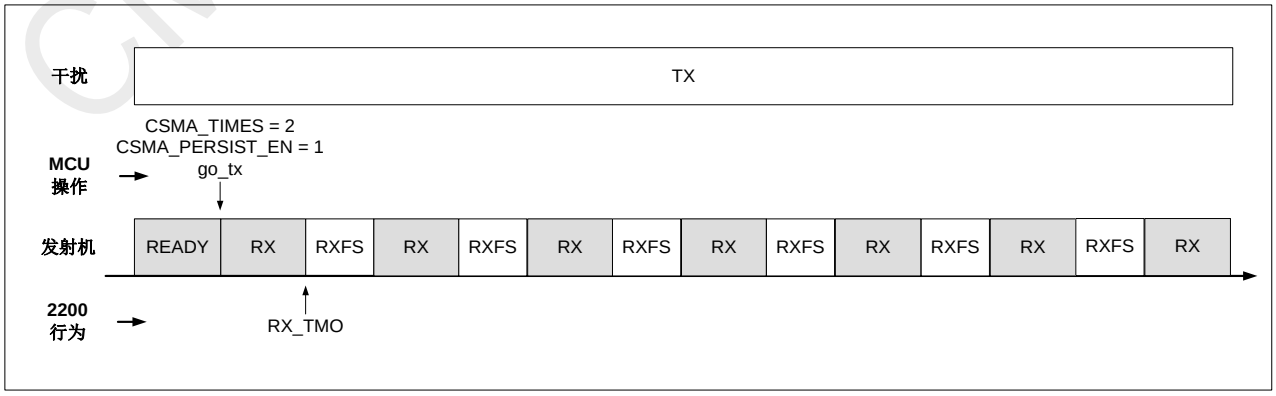
1. CSMA_PERSIST_EN = 0，达到最大侦听次数后失败退出



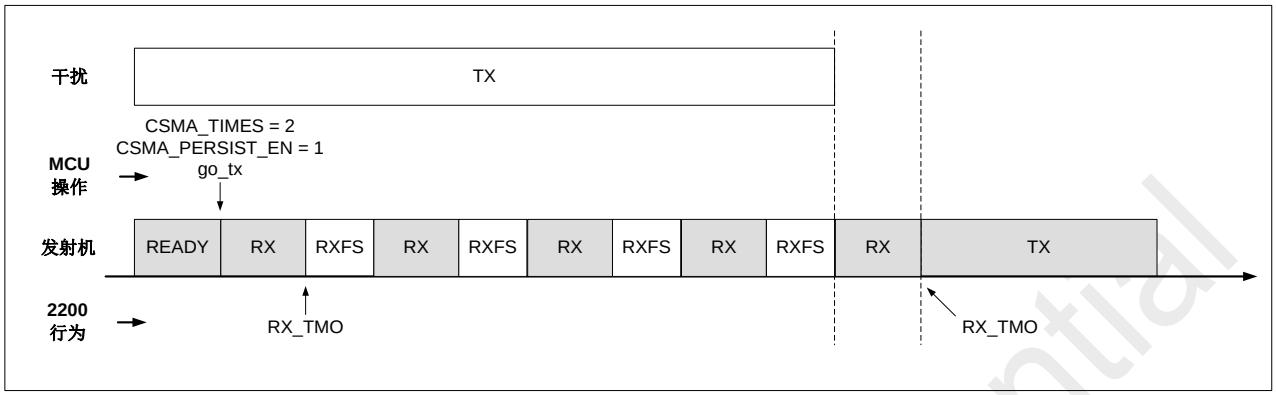
2. CSMA_PERSIST_EN = 0，未达到最大检测次数成功发射



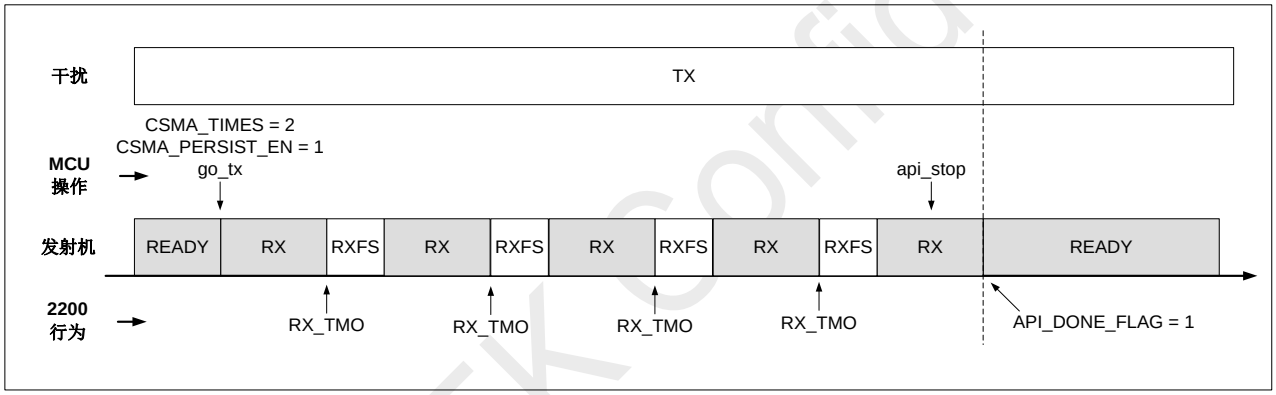
3. CSMA_PERSIST_EN = 1，信道一直处于繁忙



4. CSMA_PERSIST_EN = 1，成功等到信道空闲把数据发送出去



5. CSMA_PERSIST_EN = 1，信道一直处于繁忙，用户使用 API_STOP 退出



6. 文档变更记录

表 23. 文档变更记录表

版本号	章节	变更描述	日期
0.5	所有	初始版本发布	2020-09-17

7. 联系方式

无锡泽太微电子有限公司深圳分公司

深圳市南山区西丽街道万科云城 3 期 8 栋 A 座 30 楼

邮编： 518055

电话： +86-755-83231427

销售： sales@cmostek.com

技术支持： support@cmostek.com

网址： www.cmostek.com