RMU 通信接口定义

Release 2.8

目录：

1. 通信接口

RMU 系列超高频 RFID 读写模块通过 UART 与上位机通信。上位机（例如，PC 或单片机）需要按照规定数据格式往 RMU 发送命令并接收 RMU 返回的信息。为了方便用户在 PC 上的应用开发，我公司提供可在 WINDOWS 平台下运行的 RMU 的 API 函数库。该函数库用 C++语言编写并封装成 WINDOWS 的标准动态链接库（DLL）。

表 1-1 RMU 支持的 UART 参数

波特率	57600
数据位	8
奇偶校验	无
停止位	1

2. 数据包格式

上位机发送到 RMU 的数据包以下称“命令”，而 RMU 返回到上位机的数据包以下称“响应”。以下所有数据段的长度单位为字节。RMU 与上位机传递的数据包的通用格式见表 2-1 和表 2-2：

表 2-1 命令的数据包格式

数据段	SOF	LENGTH	CMD	PAYLOAD	*CRC-16	EOF
长度	1	1	1	< 254	2	1

表 2-2 响应的数据包格式

数据段	SOF	LENGTH	CMD	STATUS	PAYLOAD	*CRC-16	EOF
长度	1	1	1	1	< 253	2	1

注：有*为可选部分，下同。

2.1 SOF (Start Of Frame)

SOF 是一个字节的常数（SOF == 0xAA），表示数据帧的开始。

2.2 LENGTH

LENGTH 部分是按字节计算的<SOF>和<EOF>之间数据（即<LENGTH>、<CMD>、<STATUS>、<PAYLOAD>、<CRC-16>）的长度。

2.3 CMD

CMD 数据段的定义见表 2-3。

表 2-3 CMD 数据段定义

位	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
描述	CRC 控制位	RMU 命令						
功能	0 = 数据包中没有 CRC-16 1 = 数据包中带有 CRC-16	见表 2-6						

用户可利用 CMD 字节的 Bit 7 选择是否使用数据包的 CRC-16 验证功能。RMU 返回的响应的 CRC-16 设置与上位机的相应命令保持一致。读写器命令见表 2-5。

2.4 STATUS

STATUS 是 RMU 的响应中包含的对上位机命令的执行状态。STATUS 只在 RMU 的响应中，上位机的命令中没有 STATUS 部分。STATUS 中高四位是通用的标志位，而低四位是各命令中特有的状态。STATUS 通用标志位的定义见表 2-4，低四位的定义详见各命令的状态定义表。

表 2-4 STATUS 的通用标志位定义

位	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3 ~ 0
描述	1 = 执行命令失败 0 = 执行命令成功	1 = CRC 验证失败 0 = CRC 验证成功	保留	保留	见各命令的状态定义表

2.5 PAYLOAD

PAYLOAD 是需要传递的实际数据。除了在各命令格式中已定义的 PAYLOAD 有效字节外，在 LENGTH 可表示的范围内可延长任意 PAYLOAD，RMU 不对其进行操作。

2.6 CRC-16

CRC-16 部分是对<LENGTH>、<CMD>、<STATUS>(响应中)和<PAYLOAD>部分计算的 CRC-16 值。用户可通过 CMD 的 Bit 7 选择是否使用该选项。

当上位机命令的 CRC-16 验证失败时 RMU 返回固定格式的响应，其格式如表 2-5，其中 STATUS 字节的值为 0xC0。

表 2-5 CRC 验证失败响应

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

2.7 EOF (End Of Frame)

EOF 是一个字节的常数（EOF == 0x55），表示数据帧的结束。

2.8 RMU 命令列表

表 2-6 RMU 命令列表

命令	**值（hex）**	**功能**	**响应等待时间（ms）**
RMU_GET_STATUS	00	询问状态	200
RMU_GET_POWER	01	读取功率设置	200
RMU_SET_POWER	02	设置功率	200
RMU_GET_FRE	05	读取频率设置	200
RMU_SET_FRE	06	设置频率	200
RMU_GET_VERSION	07	读取 RMU 信息	200
RMU_GET_ANT	08	读取天线设置	200
RMU_SET_ANT	09	设置天线	200
RMU_INVENTORY	10	识别标签（单标签识别）	200
RMU_INVENTORY_ANTI	11	识别标签（防碰撞识别）	200
RMU_STOP_GET	12	停止识别标签	200
RMU_READ_DATA	13	读取标签数据	200
RMU_WRITE_DATA	14	写入标签数据	200
RMU_ERASE_DATA	15	擦除标签数据	200

RMU_LOCK_MEM	16	锁定标签	200
RMU_KILL_TAG	17	销毁标签	200
RMU_INVENTORY_SINGLE	18	识别标签（单步识别）	200
RMU_WIEGAND_INVENTORY	19	韦根识别	200
RMU_SINGLE_READ_DATA	20	读取标签数据（不指定 UII）	200
RMU_SINGLE_WRITE_DATA	21	写入标签数据（不指定 UII）	200

2.9 插入字节

为了避免数据中出现 SOF、EOF 字节，实际通信过程中利用插入字节保证 SOF 和 EOF 的唯一性。当发送数据包的 SOF 和 EOF 之间出现 0xAA、0x55、0xFF 字节时，发送方应在该字节前插入一个 0xFF 字节。接收方接收到包含插入字节的数据后应删除插入字节并提取有效数据。插入字节不计入 LENGTH。插入字节的实例参考附录 E。

2.10 RMU 的响应时间

上位机发送命令后当 RMU 在一定时间内没有响应，则说明命令格式不正确或 RMU 在命令执行过程中遇到不可预测的错误。这时上位机可再次发送命令。各命令的响应等待时间见表 2-6。

3. 命令定义

3.1. 询问状态

3.1.1 功能简介

该命令询问 RMU 的状态，正确接收该命令之后 RMU 回复功放的开关状态。用户可利用该命令完成两种操作：

1）在操作 RMU 的过程中，查询 RMU 的功放是否打开，并执行相关操作。

2）查询 RMU 是否连接，如果有响应则说明 RMU 已经连接，而如果在指定时间内没有响应则说明 RMU 不可达。

3.1.2 数据格式

表 3-1-1 询问状态命令格式

数据段	SOF	LEN	CMD	*CRC	EOF
长度	1	1	1	2	1

表 3-1-2 询问状态响应格式

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

3.1.3 命令状态定义

表 3-1-3 询问状态 STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 功放状态为开启 0 = 功放状态为关闭

注：该命令的 STATUS Bit0 只在 Bit 7 为 0 时有效。

3.1.4 相关 API 函数

表 3-1-4 询问状态相关 API 函数

函数名	说明
RmuOpenAndConnect()	打开 COM 端口并连接 RMU
RmuGetPaStatus()	询问功放状态

3.1.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 02 00 55	成功：aa 03 00 00 55
	失败：无返回

3.2. 读取功率设置

3.2.1 功能简介

该命令读取 RMU 的功率设置。用户使用 RMU 对标签进行操作前可用该命令读取 RMU 的功率设置。该命令有两种响应格式，即操作成功（表 3-2-2）和失败（表 3-2-3）。

3.2.2 数据格式

表 3-2-1 读取功率设置命令格式

数据段	SOF	LEN	CMD	*CRC	EOF
长度	1	1	1	2	1

表 3-2-2 读取功率设置响应格式（成功）

数据段	SOF	LEN	CMD	STATUS	POWER	*CRC	EOF
长度	1	1	1	1	1	2	1

表 3-2-3 读取功率设置响应格式（失败）

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

表 3-2-4 POWER 数据段格式

POWER	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
描述	功放控制模式	输出功率 （dbm）						
功能	0 = 手动控制，用户需要使用“开启功放”和“关闭功放”命令控制功放 1 = 自动控制，RMU 发射命令时自动开启功放，发射完后自动关闭功放							

3.2.3 命令状态定义

该命令只支持通用状态位。

3.2.4 相关 API 函数

表 3-2-5 读取功率设置相关 API 函数

函数名	说明
RmuGetPower()	读取 RMU 的功率设置

3.2.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 02 01 55	成功：aa 04 01 00 9a 55
	失败：无返回

3.3. 设置功率

3.3.1 功能简介

该命令设置 RMU 的输出功率和功放控制模式。用户使用 RMU 对标签进行操作前需要用该命令设置 RMU 的输出功率和功放控制模式。若用户没有设置 RMU 的功率，RMU 工作时将使用默认设置。该命令根据 OPTION 数据段的值可单独设置功放控制模式或输出功率，也可同时设置。

3.3.2 数据格式

表 3-3-1 设置功率命令格式

数据段	SOF	LEN	CMD	OPTION	POWER	*CRC	EOF
长度	1	1	1	1	1	2	1

表 3-3-2 设置功率响应格式

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

表 3-3-3 OPTION 数据段格式

OPTION	Bit 7 ~ 2	Bit 1	Bit 0
描述	保留	设置功放控制模式控制位	设置输出功率控制位
功能	保留	1：POWER 的 Bit7 有效 0：POWER 的 Bit7 无效	1：POWER 的 Bit6~0 有效 0：POWER 的 Bit6~0 无效

注：POWER 数据段的定义见表 3-2-3。

3.3.3 命令状态定义

该命令只支持通用状态位。

3.3.4 相关 API 函数

表 3-3-4 设置功率相关 API 函数

函数名	说明
RmuSetPower()	设置 RMU 的功率

3.3.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 04 02 03 9a 55	成功：aa 03 02 00 55
	失败：无返回

3.4. 读取频率设置

3.4.1 功能简介

该命令读取 RMU 的频率设置。频率设置相关数据段的说明见 3.5 节。

注：目前只有 RMU900+支持该命令。

3.4.2 数据格式

表 3-6-1 读取频率设置命令格式

数据段	SOF	LEN	CMD	*CRC	EOF
长度	1	1	1	2	1

表 3-6-2 读取频率设置响应格式

数据段	SOF	LEN	CMD	STATUS	BF	CN	SPC	*CRC	EOF
长度	1	1	1	1	2	1	1	2	1

3.4.3 命令状态定义

该命令只支持通用状态位。

3.4.4 相关 API 函数

表 3-6-3 读取频率设置相关 API 函数

函数名	说明
RmuGetFrequency()	读取 RMU 频率设置

3.4.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 02 05 55	成功：aa 07 05 00 03 48 1f e8 55
	失败：无返回

3.5. 设置频率

3.5.1 功能简介

该命令设置 RMU 的频率。RMU 的频率设置有三个参数：起始频率（BF）、频道数（CN）和频道带宽（SPC）。其中起始频率是 RMU 使用的最低频率，频道数是 RMU 在跳频时支持的最大频道个数，频道带宽是每一频道的信道带宽。

RMU 支持的基准频率范围为 860Mhz~960Mhz，用户可以依据应用环境需求，自己定义频率范围。目前 RMU 允许使用四种频率设置模式：

(1)、“Chinese 标准”模式，该模式采用中国目前规定 RFID 使用频段 840-845MHz、920-925MHz。关于中国 RFID 使用频段相关说明请查看【附录 F】。

(2)、“ETSI 标准”模式，该模式采用欧洲标准，有效频率范围为 865-868MHz。关于 ETSI 标准相关说明请查看【附录 F】。

(3)、“定频”模式，该模式允许用户自定义频率，取值范围为 840-960。

(4)、“用户自定义”模式，该模式支持跳频。用户需要设置三个参数：起始频率（BF）、频道带宽（SPC）、频道数（CN）。

依据表 3-5-2 和表 3-5-3，起始频率（BF）、频道数（CN）、频道带宽（SPC）、最终频率和带宽存在以下关系：

(1)、起始频率（BF）=【起始频率基数】＋ 【频率基数】×【起始频率尾数积数】

如：起始频率（BF）＝840MHz ＋ 125KHZ × 5 ＝ 840.625MHz

(2)、频道带宽（SPC）＝【频道带宽积数】×【频率基数】

如：频道带宽（SPC）＝2 × 125KHz ＝ 250KHz

(3)、最终频率 ＝ 起始频率（BF）＋ （频道数（CN）－1）× 频道带宽（SPC）

如：最终频率 ＝ 840.625MHz ＋ （16－1） × 250KHz ＝ 844.375MHz

(3)、带宽 ＝ 最终频率 － 起始频率（BF）

如：带宽 ＝ 844.375MHz － 840.625MHz ＝ 3.75MHz

注：目前只有 RMU900+支持该命令。【频道带宽】只能取 125KHz、250KHz、500KHz、200KHz、400KHz 和 600KHz；【频率基数】×【起始频率尾数积数】不能超过 1000KHz；当【频率基数】为 50KHz 时，【带宽】不能大于 12MHz，当【频率基数】为 125KHz 时，【带宽】不能大于 32MHz。

3.5.2 数据格式

表 3-5-1 设置频率命令格式

数据段	SOF	LEN	CMD	BF	CN	SPC	*CRC	EOF
长度	1	1	1	2	1	1	2	1

表 3-5-2 SPC 字段定义

位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
功能	0：顺序跳频 1：随机跳频	00：Chines 标准 01：ETSI 标准 10：定频 11：用户自定义		频率基数： 0：50KHz 1：125KHz	频道带宽积数			

表 3-5-3 BF 字段定义

位	Bit15～Bit10	Bit9~Bit0
功能	起始频率尾数积数	起始频率基数

表 3-5-4 设置频率响应格式

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

3.5.3 命令状态定义

该命令只支持通用状态位。

3.5.4 相关 API 函数

表 3-5-3 设置频率相关 API 函数

函数名	说明
RmuSetFrequency()	设置 RMU 频率

3.5.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 06 06 03 48 1f e8 55	成功：aa 03 06 00 55
	失败：无返回

3.6. 识别标签（单标签识别）

3.6.1 功能简介

该命令启动标签识别循环，对单张标签进行识别时使用该命令。该命令有两种响应格式：RMU 接收该命令后返回识别标签响应（表 3-6-2）告诉上位机启动标签识别循环成功与否；若启动标签识别循环成功，RMU 连续返回获取标签号响应（表 3-6-3）直到接收停止识别标签命令，每个获取标签号响应只返回一张标签的 UII。

3.6.2 数据格式

表 3-6-1 识别标签命令格式（单标签识别）

数据段	SOF	LEN	CMD	*CRC	EOF
长度	1	1	1	2	1

表 3-6-2 识别标签响应格式

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

表 3-6-3 获取标签号响应格式

数据段	SOF	LEN	CMD	STATUS	UII	*CRC	EOF
长度	1	1	1	1		2	1

注：本文档中的 UII 包括 PC bits，即 PC+UII。UII 的格式见附录 A。

3.6.3 命令状态定义

表 3-6-4 识别标签 STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 识别标签响应（不包含 UII） 0 = 获取标签号响应（包含 UII）

注：该命令的 STATUS Bit0 只在 Bit 7 为 0 时有效。

3.6.4 相关 API 函数

表 3-6-5 识别标签相关 API 函数

函数名	说明
RmuInventory()	获取标签 UII

3.6.5 命令示例

发送命令格式（hex）	返回数据格式（hex）	
AA 02 10 55	成功：	先返回确认命令：aa 03 10 01 55（收到识别标签命令）
		再返回标签数据：aa 07 10 00 08 00 00 01 55（不断返回标签数据）
	失败：仅返回确认命令：aa 03 10 01 55（没有识别到标签）	

3.7. 识别标签（防碰撞识别）

3.7.1 功能简介

该命令启动标签识别循环，对多张标签进行识别时使用该命令。发送命令时需指定防碰撞识别的初始 Q 值。若 Q 设为 0，RMU 使用默认 Q 值。该命令的响应方式与单标签识别命令一致。

3.7.2 数据格式

表 3-7-1 识别标签命令格式（防碰撞识别）

数据段	SOF	LEN	CMD	Q	*CRC	EOF
长度	1	1	1	1	2	1

表 3-7-2 Q 数据段格式

Q	Bit 7 ~ Bit 4	Bit 3 ~ Bit 0
描述	保留	Q Bit 3 ~ 0

表 3-7-3 识别标签响应格式

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

表 3-7-4 获取标签号响应格式

数据段	SOF	LEN	CMD	STATUS	UII	*CRC	EOF
长度	1	1	1	1		2	1

3.7.3 命令状态定义

表 3-7-5 识别标签 STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 识别标签响应（不包含 UII） 0 = 获取标签号响应（包含 UII）

注：该命令的 STATUS Bit0 只在 Bit 7 为 0 时有效。

3.7.4 相关 API 函数

表 3-7-6 识别标签相关 API 函数

函数名	说明
RmuInventory()	获取标签 UII

3.7.5 命令示例

发送命令格式（hex）	返回数据格式（hex）	
aa 03 11 03 55	成功：	先返回确认命令：aa 03 11 01 55（收到识别标签命令）
		再返回标签数据：aa 07 11 00 08 00 00 01 55（不断返回标签数据）
	失败：仅返回确认命令：aa 03 11 01 55（没有识别到标签）	

3.8. 识别标签（单步识别）

3.8.1 功能简介

该命令识别单张标签。与单标签识别和防碰撞识别命令不同的是：该命令不启动识别循环。每次上位机发送该命令时，RMU 识别标签，如果识别到标签则返回标签号，若没有识别到标签则无返回。

3.8.2 数据格式

表 3-8-1 识别标签命令格式（单步识别）

数据段	SOF	LEN	CMD	*CRC	EOF
长度	1	1	1	2	1

表 3-8-2 识别标签响应格式

数据段	SOF	LEN	CMD	STATUS	UII	*CRC	EOF
长度	1	1	1	1		2	1

3.8.3 命令状态定义

该命令只支持通用状态位。

3.8.4 相关 API 函数

表 3-8-3 识别标签相关 API 函数

函数名	说明
RmuInventorySingle()	单步获取标签 UII

3.8.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 02 18 55	成功：aa 07 18 00 08 00 00 01 55
	失败：aa 03 18 01 55

3.9. 韦根识别

3.9.1 功能简介

用于识别单张电子标签，通过韦根接口输出数据,符合 wiegend-26 协议。不启动防碰撞功能。当上位机与 RMU 900 读写器连接成功后，该功能才能进行操作。

[注]韦根接口数据输出时，需外部提供 12v 电源到韦根接口（12v 引脚与 GND 引脚之间）

每次上位机发送该命令时，RMU 识别标签，如果识别到标签则返回标签号，若没有识别到标签则无返回。

3.9.2 数据格式

表 3-9-1 识别标签命令格式

数据段	SOF	LEN	CMD	*CRC	EOF
长度	1	1	1	2	1

表 3-9-2 识别标签响应格式

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

表 3-9-3 获取标签号响应格式

数据段	SOF	LEN	CMD	STATUS	UII	*CRC	EOF
长度	1	1	1	1		2	1

注：本文档中的 UII 包括 PC bits，即 PC+UII。UII 的格式见附录 A。

3.9.3 命令状态定义

表 3-9-4 识别标签 STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 识别标签响应（不包含 UII） 0 = 获取标签号响应（包含 UII）

注：该命令的 STATUS Bit0 只在 Bit 7 为 0 时有效。

3.9.4 相关 API 函数

表 3-9-5 韦根识别标签相关 API 函数

函数名	说明
RmuWiegandInventory ()	获取标签 UII（韦根识别）

3.9.5 命令示例

发送命令格式（hex）	返回数据格式（hex）	
aa 02 19 55	成功：	先返回确认命令：aa 03 19 01 55（收到识别标签命令）
		再返回标签数据：aa 07 19 00 08 00 00 01 55（不断返回标签数据）
	失败：仅返回确认命令：aa 03 19 01 55（没有识别到标签）	

3.10. 停止识别标签

3.10.1 功能简介

该命令停止识别标签 UII。RMU 接收到该命令之后跳出识别循环并不再返回标签 UII。

3.10.2 数据格式

表 3-10-1 停止识别标签命令格式

数据段	SOF	LEN	CMD	*CRC	EOF
长度	1	1	1	2	1

表 3-10-2 停止识别标签响应格式

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

3.10.3 命令状态定义

该命令只支持通用状态位。

3.10.4 相关 API 函数

表 3-10-3 停止识别标签相关 API 函数

函数名	说明
RmuStopGet()	停止识别标签 UII

3.10.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 02 12 55	成功：aa 03 12 00 55
	失败：无返回

3.11. 读取标签数据(指定 UII 模式)

3.11.1 功能简介

该命令从标签读取数据。用户需指定欲读取数据的标签的 UII 信息,方能从该电子标签内读取标签数据。该命令成功、失败时响应格式有所不同，详见表 3-11-1 及 3-11-2。

读取标签数据、写入标签数据、擦除标签数据和锁定标签操作的命令中含有标签的 ACCESS 密码（APWD 数据段），当 APWD 不全为零时 RMU 利用 ACCESS 命令确保标签处在 SECURED 状态后进行相应操作。

进行数据操作（读取标签数据、写入标签数据、擦除标签数据、锁定标签、销毁标签）时标签有可能返回错误码（Error Code）,这时 RMU 的响应中含有一个字节的错误码（可选项）。错误码的定义见附录 C。

3.11.2 数据格式

表 3-11-1 读取标签数据命令格式

数据段	SOF	LEN	CMD	APWD	BANK	PTR	CNT	UII	*CRC	EOF
长度	1	1	1	4	1	EBV	1		2	1

注 1：APWD 数据段是标签的 ACCESS PASSWORD，下同。

注 2：PTR 数据段是 EBV 格式，EBV 格式见附录 B。

注 3：CNT 数据段是以 WORD（2 字节）为单位的读出数据的长度。

表 3-11-2 读取标签数据响应格式（成功）

数据段	SOF	LEN	CMD	STATUS	DATA	*CRC	EOF
长度	1	1	1	1	CNT*2	2	1

表 3-11-3 读取标签数据响应格式（失败）

数据段	SOF	LEN	CMD	STATUS	*ECODE	*CRC	EOF
长度	1	1	1	1	1	2	1

注：ECODE（Error Code）数据段是可选项，下同。

3.11.3 命令状态定义

表 3-11-4 读取标签数据 STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 响应中含 ECODE 数据段 0 = 响应中不含 ECODE 数据段

注：该命令的 STATUS Bit0 只在 Bit 7 为 1 时有效。

3.11.4 相关 API 函数

表 3-11-5 读取标签数据相关 API 函数

函数名	说明
RmuReadData()	读取标签数据

3.11.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 0c 13 00 00 00 00 01 01 01 08 00 00 01 55	成功：aa 05 13 00 08 00 55
	失败：aa 04 13 81 04 55

3.12. 读取标签数据（不指定 UII 模式）

3.12.1 功能简介

该命令从标签读取数据。用户无须指定电子标签的 UII 即可从该电子标签内读取指定存储空间的数据信息,并返回该电子标签的 UII 信息。

读取标签数据、写入标签数据、擦除标签数据和锁定标签操作的命令中含有标签的 ACCESS 密码（APWD 数据段），当 APWD 不全为零时 RMU 利用 ACCESS 命令确保标签处在 SECURED 状态后进行相应操作。

进行数据操作（读取标签数据、写入标签数据、擦除标签数据、锁定标签、销毁标签）时标签有可能返回错误码（Error Code）,这时 RMU 的响应中含有一个字节的错误码（可选项）。错误码的定义见附录 C。

3.12.2 数据格式

表 3-12-1 读取标签数据（不指定 UII）命令格式

数据段	SOF	LEN	CMD	APWD	BANK	PTR	CNT	*CRC	EOF
长度	1	1	1	4	1	EBV	1	2	1

注 1：APWD 数据段是标签的 ACCESS PASSWORD，下同。

注 2：PTR 数据段是 EBV 格式，EBV 格式见附录 B。

注 3：CNT 数据段是以 WORD（2 字节）为单位的读出数据的长度。

表 3-12-2 读取标签数据（不指定 UII）响应格式（成功）

数据段	SOF	LEN	CMD	STATUS	DATA	UII	*CRC	EOF
长度	1	1	1	1	CNT*2		2	1

表 3-12-3 读取标签数据（不指定 UII）响应格式（失败）

数据段	SOF	LEN	CMD	STATUS	*ECODE	*CRC	EOF
长度	1	1	1	1	1	2	1

注：ECODE（Error Code）数据段是可选项，下同。

3.12.3 命令状态定义

表 3-12-4 读取标签数据（不指定 UII）STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 响应中含 ECODE 数据段 0 = 响应中不含 ECODE 数据段

注：该命令的 STATUS Bit0 只在 Bit 7 为 1 时有效。

3.12.4 相关 API 函数

表 3-12-5 读取标签数据（不指定 UII）相关 API 函数

函数名	说明
RmuReadDataSingle()	读取标签数据（不指定 UII）

3.12.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 09 20 00 00 00 00 01 01 01 55	成功：aa 09 20 00 08 00 08 00 00 01 55
	失败：aa 04 20 81 04 55

3.13. 写入标签数据(指定 UII)

3.13.1 功能简介

该命令往标签写入数据。在这种写入方式下，用户需指定欲写入数据的电子标签的 UII 信息。

3.13.2 数据格式

表 3-13-1 写入标签数据命令格式

数据段	SOF	LEN	CMD	APWD	BANK	PTR	CNT	DATA	UII	*CRC	EOF
长度	1	1	1	4	1	EBV	1	CNT*2		2	1

注：CNT 数据段是以 WORD（2 字节）为单位的 DATA 的长度。现只支持 CNT 为 1。

表 3-13-2 写入标签数据响应格式

数据段	SOF	LEN	CMD	STATUS	*ECODE	*CRC	EOF
长度	1	1	1	1	1	2	1

3.13.3 命令状态定义

表 3-13-3 写入标签数据 STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 响应中含 ECODE 数据段 0 = 响应中不含 ECODE 数据段

注：该命令的 STATUS Bit0 只在 Bit 7 为 1 时有效。

3.13.4 相关 API 函数

表 3-13-4 写入标签数据相关 API 函数

函数名	说明
RmuWriteData()	写入标签数据

3.13.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 0f 14 00 00 00 00 01 01 01 10 00 08 00 00 01 55	成功：aa 03 14 00 55
	失败：aa 04 14 81 04 55

3.14. 写入标签数据（不指定 UII）

3.14.1 功能简介

该命令往标签写入数据。用户无须指定电子标签的 UII 即可向该电子标签的指定地址的存储空间写入数据信息,并返回该电子标签的 UII 信息。

3.14.2 数据格式

表 3-14-1 写入标签数（不指定 UII）据命令格式

数据段	SOF	LEN	CMD	APWD	BANK	PTR	CNT	DATA	*CRC	EOF
长度	1	1	1	4	1	EBV	1	CNT*2	2	1

注：CNT 数据段是以 WORD（2 字节）为单位的 DATA 的长度。现只支持 CNT 为 1。

表 3-14-2 写入标签数据（不指定 UII）响应格式

数据段	SOF	LEN	CMD	STATUS	UII	*ECODE	*CRC	EOF
长度	1	1	1	1		1	2	1

3.14.3 命令状态定义

表 3-14-3 写入标签数据（不指定 UII）STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 响应中含 ECODE 数据段，不含 UII 数据 0 = 响应中不含 ECODE 数据段，不含 UII 数据

注：该命令的 STATUS Bit0 只在 Bit 7 为 1 时有效。

3.14.4 相关 API 函数

表 3-14-4 写入标签数据（不指定 UII）相关 API 函数

函数名	说明
RmuWriteDataSingle()	写入标签数据（不指定 UII）

3.14.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 0b 21 00 00 00 00 01 01 01 10 00 55	成功：aa 07 21 00 08 00 00 01 55
	失败：aa 04 21 81 04 55

3.15. 擦除标签数据

3.15.1 功能简介

该命令擦除指定标签的指定数据段。用户获取标签号后可用该命令擦除标签数据。该命令只对支持 BlockErase 命令的标签有效。

3.15.2 数据格式

表 3-15-1 擦除标签数据命令格式

数据段	SOF	LEN	CMD	APWD	BANK	PTR	CNT	UII	*CRC	EOF
长度	1	1	1	4	1	EBV	1		2	1

注：CNT 数据段是以 WORD（2 字节）为单位的需要擦除的数据长度。

表 3-15-2 擦除标签数据响应格式

数据段	SOF	LEN	CMD	STATUS	*ECODE	*CRC	EOF
长度	1	1	1	1	1	2	1

3.15.3 命令状态定义

表 3-15-3 擦除标签数据 STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 响应中含 ECODE 数据段 0 = 响应中不含 ECODE 数据段

注：该命令的 STATUS Bit0 只在 Bit 7 为 1 时有效。

3.15.4 相关 API 函数

表 3-15-4 擦除标签数据相关 API 函数

函数名	说明
RmuEraseData()	擦除标签数据

3.15.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 0d 15 00 00 00 00 11 01 01 08 00 00 01 55	成功：aa 03 15 00 55
	失败：aa 04 15 81 04 55

3.16. 锁定标签

3.16.1 功能简介

该命令对指定标签的指定数据区进行 LOCK 操作。用户获取标签号后可用该命令对标签进行 LOCK 操作。

3.16.2 数据格式

表 3-16-1 锁定标签命令格式

数据段	SOF	LEN	CMD	APWD	LOCKDATA	UII	*CRC	EOF
长度	1	1	1	4	3		2	1

注：LOCKDATA 数据段的高四位为保留位，低二十位是 Lock-Command Payload，详见附录 D。

表 3-16-2 锁定标签响应格式

数据段	SOF	LEN	CMD	STATUS	*ECODE	*CRC	EOF
长度	1	1	1	1	1	2	1

3.16.3 命令状态定义

表 3-16-3 锁定标签 STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 响应中含 ECODE 数据段 0 = 响应中不含 ECODE 数据段

注：该命令的 STATUS Bit0 只在 Bit 7 为 1 时有效。

3.16.4 相关 API 函数

表 3-16-4 锁定标签相关 API 函数

函数名	说明
RmuLockMem()	锁定标签

3.16.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 0d 16 00 00 00 00 00 10 04 08 00 00 01 55	成功：aa 03 16 00 55
	失败：aa 04 16 81 04 55

3.17. 销毁标签

3.17.1 功能简介

该命令销毁指定标签。用户获取标签号后可用该命令销毁指定标签。

3.17.2 数据格式

表 3-17-1 销毁标签命令定义

数据段	SOF	LEN	CMD	KILLPWD	UII	*CRC	EOF
长度	1	1	1	4		2	1

注：KILLPWD 数据段时 4 个字节的 Kill Password。

表 3-17-2 销毁标签响应定义

数据段	SOF	LEN	CMD	STATUS	*ECODE	*CRC	EOF
长度	1	1	1	1	1	2	1

3.17.3 命令状态定义

表 3-17-3 销毁标签 STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 响应中含 ECODE 数据段 0 = 响应中不含 ECODE 数据段

注：该命令的 STATUS Bit0 只在 Bit 7 为 1 时有效。

3.17.4 相关 API 函数

表 3-17-4 销毁标签相关 API 函数

函数名	说明
RmuKillTag()	销毁标签

3.17.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 0a 17 00 00 00 00 08 00 00 01 55	成功：aa 03 17 00 55
	失败：aa 04 17 81 04 55

3.18. 读取 RMU 信息

3.18.1 功能简介

该命令读取 RMU 的硬件序列号和软件版本号。其中，RMU 的硬件序列号是 6 个字节的十六进制数，软件版本号是一个字节。软件版本字节的前四个比特是软件的主版本号，后四个比特是次版本号。

3.18.2 数据格式

表 3-18-1 读取 RMU 信息命令定义

数据段	SOF	LEN	CMD	*CRC	EOF
长度	1	1	1	2	1

表 3-18-2 读取 RMU 信息响应定义

数据段	SOF	LEN	CMD	STATUS	SERIAL	VERSION	*CRC	EOF
长度	1	1	1	1	6	1	2	1

3.18.3 命令状态定义

表 3-18-3 读取 RMU 信息 STATUS

位	Bit 7 ~ 4	Bit 3 ~ 1	Bit 0
功能	通用位	保留	1 = 该 RMU 没有定义相关信息 0 = 成功读取 RMU 信息

注：该命令的 STATUS Bit0 只在 Bit 7 为 1 时有效。

3.18.4 相关 API 函数

表 3-18-4 销毁标签相关 API 函数

函数名	说明
RmuGetVersion()	读取 RMU 信息

3.18.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 02 07 55	成功：aa 0a 07 01 ff ff ff ff ff ff ff ff ff ff ff ff ff ff 55
	失败：无返回

3.19. 读取天线设置

3.19.1 功能简介

该命令读取 RMU 的天线设置。用户使用 RMU 对标签进行操作前可用该命令读取 RMU 的天线设置。该命令有两种响应格式，即操作成功（表 3-19-2）和失败（表 3-19-3）。

注：目前只有 RMU900E 支持该命令。

3.19.2 数据格式

表 3-19-1 读取天线设置命令格式

数据段	SOF	LEN	CMD	*CRC	EOF
长度	1	1	1	2	1

表 3-19-2 读取天线设置响应格式（成功）

数据段	SOF	LEN	CMD	STATUS	ANT	*CRC	EOF
长度	1	1	1	1	1	2	1

表 3-19-3 读取天线设置响应格式（失败）

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

表 3-19-4 ANT 数据段格式

ANT	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
功能	保留						天线选择	
描述							00：天线 1 01：天线 2 10：天线 3 11：天线 4	

3.19.3 命令状态定义

该命令只支持通用状态位。

3.19.4 相关 API 函数

表 3-19-5 读取天线设置相关 API 函数

函数名	说明
RmuGetAntenna()	读取 RMU 的天线设置

3.21.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 02 08 55	成功：aa 04 08 00 01 55
	失败：无返回

3.20. 设置天线

3.20.1 功能简介

该命令设置 RMU 的天线。用户使用 RMU 对标签进行操作前需要用该命令设置 RMU 的天线。若用户没有设置 RMU 的天线，RMU 工作时将使用默认设置（天线 1）。

注：目前只有 RMU900E 支持该命令。

3.20.2 数据格式

表 3-20-1 设置天线命令格式

数据段	SOF	LEN	CMD	ANT	*CRC	EOF
长度	1	1	1	1	2	1

表 3-20-2 设置天线响应格式

数据段	SOF	LEN	CMD	STATUS	*CRC	EOF
长度	1	1	1	1	2	1

3.20.3 命令状态定义

该命令只支持通用状态位。

3.20.4 相关 API 函数

表 3-20-4 设置天线相关 API 函数

函数名	说明
RmuSetAntenna()	设置 RMU 的天线

3.20.5 命令示例

发送命令格式（hex）	返回数据格式（hex）
aa 03 09 02 55	成功：aa 03 09 00 55
	失败：无返回

4. API 函数定义

4.1 RmuOpenAndConnect ()

4.1.1 功能简介

该函数打开串口并与 RMU 建立链接。

4.1.2 函数原型

int WINAPI RmuOpenAndConnect (HANDLE &hCom, char* cPort, UCHAR flagCrc);

4.1.3 返回值

1：端口打开成功并连接到 RMU。

其他：打开端口或连接 RMU 失败。

4.1.4 输入参数

HANDLE &hCom：通信端口句柄，初始化为 NULL。

char* cPort：串口，例如 COM1、COM2 等。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.2 RmuCloseAndDisconnect ()

4.2.1 功能简介

该函数关闭 RMU 的功放并关闭通信端口。

4.2.2 函数原型

int WINAPI RmuCloseAndDisconnect (HANDLE &hCom, UCHAR flagCrc);

4.2.3 返回值

1：关闭 RMU 功放和关闭端口成功。

其他：关闭 RMU 功放或关闭端口失败。

4.2.4 输入参数

HANDLE &hCom：通信端口句柄。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.3 RmuGetPaStatus ()

4.3.1 功能简介

该函数读取 RMU 功放状态。

4.3.2 函数原型

int WINAPI RmuGetPaStatus (HANDLE hCom, UCHAR* uStatus, UCHAR flagCrc);

4.3.3 返回值

1：成功读取 RMU 功放状态。

其他：读取 RMU 功放状态失败。

4.3.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uStatus：RMU 的功放状态。1：功放状态为开启；0：功放状态为关闭。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.4 RmuGetPower ()

4.4.1 功能简介

该函数读取 RMU 的功率设置。

4.4.2 函数原型

int WINAPI RmuGetPower (HANDLE hCom, UCHAR* uPower, UCHAR flagCrc);

4.4.3 返回值

1：成功读取 RMU 的功率设置。

其他：读取 RMU 的功率设置失败。

4.4.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uPower：RMU 返回的 POWER 字节。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.5 RmuSetPower ()

4.5.1 功能简介

该函数设置 RMU 的功率。

4.5.2 函数原型

int WINAPI RmuSetPower (HANDLE hCom, UCHAR uOption, UCHAR uPower, UCHAR flagCrc);

4.5.3 返回值

1：成功设置 RMU 的功率。

其他：设置 RMU 的功率失败。

4.5.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR uOption：设置功率命令的 OPTION 字节。

UCHAR uPower：设置功率命令的 POWER 字节。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.6 RmuGetFrequency ()

4.6.1 功能简介

该函数读取 RMU 的频率设置。

4.6.2 函数原型

int WINAPI RmuGetFrequency (HANDLE hCom, UCHAR* uBaseFre, UCHAR* uChannNum, UCHAR* uChannSpc, UCHAR flagCrc);

4.6.3 返回值

1：成功读取 RMU 的频率设置。

其他：读取 RMU 的频率设置失败。

4.6.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uBaseFre：RMU 返回的 BF 字节（基准频率，2 字节）。

UCHAR* uChannNum：RMU 返回的 CN 字节（频道数，1 字节）。

UCHAR* uChannSpc：RMU 返回的 SPC 字节（频道带宽，1 字节）。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.7 RmuSetFrequency ()

4.7.1 功能简介

该函数设置 RMU 的频率。

4.7.2 函数原型

int WINAPI RmuSetFrequency(HANDLE hCom, UCHAR* uBaseFre, UCHAR uChannNum, UCHAR uChannSpc, UCHAR flagCrc);

4.7.3 返回值

1：成功设置 RMU 的频率。

其他：设置 RMU 的频率失败。

4.7.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uBaseFre：RMU 的 BF 字节（基准频率，2 字节）。

UCHAR* uChannNum：RMU 的 CN 字节（频道数，1 字节）。

UCHAR* uChannSpc：RMU 的 SPC 字节（频道带宽，1 字节）。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.8 RmuInventory ()

4.8.1 功能简介

该函数启动 RMU 的识别循环。

4.8.2 函数原型

int WINAPI RmuInventory (HANDLE hCom, UCHAR flagAnti, UCHAR initQ, UCHAR flagCrc);

4.8.3 返回值

1：成功启动 RMU 的识别循环。

其他：启动 RMU 的识别循环失败。

4.8.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR flagAnti：是否使用防碰撞识别功能。1：防碰撞识别；0：单标签识别。

UCHAR initQ：防碰撞识别过程的初始 Q 值，flagAnti 为 1 时有效。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.9 RmuInventorySingle ()

4.9.1 功能简介

该函数单步识别标签号。

4.9.2 函数原型

int WINAPI RmuInventorySingle (HANDLE hCom, UCHAR* uLenUii, UCHAR* uUii , UCHAR flagCrc);

4.9.3 返回值

1：成功识别标签号。

其他：识别标签号失败。

4.9.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uLenUii：标签 UII 的长度，1 个字节。

UCHAR* uUii：标签 UII，至少 66 个字节。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.10 RmuWiegandInventory ()

4.10.1 功能简介

该函数启动 RMU 的韦根识别循环。

4.10.2 函数原型

int WINAPI **RmuWiegandInventory** (HANDLE hCom, UCHAR flagCrc);

4.10.3 返回值

1：成功启动 RMU 的识别循环。

其他：启动 RMU 的识别循环失败。

4.10.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.11 RmuGetReceived ()

4.11.1 功能简介

该函数读取 RMU 返回的标签 UII。

4.11.2 函数原型

int WINAPI RmuGetReceived (HANDLE hCom, UCHAR* uLenUii, UCHAR* uUii);

4.11.3 返回值

1：成功读取标签的 UII。

其他：读取标签的 UII 失败。

4.11.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uLenUii：标签 UII 的长度，1 个字节。

UCHAR* uUii：标签 UII，至少 66 个字节。

4.12 RmuStopGet ()

4.12.1 功能简介

该函数停止 RMU 的识别循环。

4.12.2 函数原型

int WINAPI RmuStopGet (HANDLE hCom, UCHAR flagCrc);

4.12.3 返回值

1：成功停止识别。

其他：停止识别失败。

4.12.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.13 RmuReadData ()

4.13.1 功能简介

该函数读取标签数据。

4.13.2 函数原型

int WINAPI RmuReadData (HANDLE hCom, UCHAR* uAccessPwd, UCHAR uBank, UCHAR* uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uReadData, UCHAR* uErrorCode, UCHAR flagCrc);

4.13.3 返回值

1：成功读取标签数据。

其他：读取标签数据失败。

4.13.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uAccessPwd：标签的 ACCESS PASSWORD。

UCHAR uBank：标签的数据段类型。

UCHAR* uPtr：起始地址的偏移量。

UCHAR uCnt：读取数据的长度（两个字节为单位）。

UCHAR* uUii：标签的 UII。

UCHAR* uReadData：读取的标签数据，至少为 uCnt * 2 个字节。

UCHAR* uErrorCode：标签返回的 Error Code。只在函数返回失败，且 uErrorCode 不等于 0xFF 时有效。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.14 RmuReadDataSingle ()

4.14.1 功能简介

该函数读取标签数据（不指定 UII）。

4.14.2 函数原型

int WINAPI RmuReadDataSingle (HANDLE hCom, UCHAR* uAccessPwd, UCHAR uBank, UCHAR* uPtr, UCHAR uCnt, UCHAR* uReadData, UCHAR* uUii, UCHAR* uLenUii, UCHAR* uErrorCode, UCHAR flagCrc);

4.14.3 返回值

1：成功读取标签数据。

其他：读取标签数据失败。

4.14.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uAccessPwd：标签的 ACCESS PASSWORD。

UCHAR uBank：标签的数据段类型。

UCHAR* uPtr：起始地址的偏移量。

UCHAR uCnt：读取数据的长度（两个字节为单位）。

UCHAR* uReadData：读取的标签数据，至少为 uCnt * 2 个字节。

UCHAR* uUii：返回值，写入的标签的 UII。

UCHAR* uLenUii ：返回值，UII 长度。

UCHAR* uErrorCode：标签返回的 Error Code。只在函数返回失败，且 uErrorCode 不等于 0xFF 时有效。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.15 RmuWriteData ()

4.15.1 功能简介

该函数写入标签数据。

4.15.2 函数原型

int WINAPI RmuWriteData (HANDLE hCom, UCHAR* uAccessPwd, UCHAR uBank, UCHAR* uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uWriteData, UCHAR* uErrorCode, UCHAR flagCrc);

4.15.3 返回值

1：成功写入标签数据。

其他：写入标签数据失败。

4.15.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uAccessPwd：标签的 ACCESS PASSWORD。

UCHAR uBank：标签的数据段类型。

UCHAR* uPtr：起始地址的偏移量。

UCHAR uCnt：写入数据的长度（两个字节为单位）。

UCHAR* uUii：标签的 UII。

UCHAR* uWriteData：需要写入的数据。

UCHAR* uErrorCode：标签返回的 Error Code。只在函数返回失败，且 uErrorCode 不等于 0xFF 时有效。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.16 RmuWriteDataSingle ()

4.161 功能简介

该函数写入标签数据（不指定 UII）。

4.16.2 函数原型

int WINAPI RmuWriteDataSingle (HANDLE hCom, UCHAR* uAccessPwd, UCHAR uBank, UCHAR* uPtr, UCHAR uCnt, UCHAR* uWriteData, UCHAR* uUii, UCHAR* uLenUii, UCHAR* uErrorCode, UCHAR flagCrc);

4.16.3 返回值

1：成功写入标签数据。

其他：写入标签数据失败。

4.16.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uAccessPwd：标签的 ACCESS PASSWORD。

UCHAR uBank：标签的数据段类型。

UCHAR* uPtr：起始地址的偏移量。

UCHAR uCnt：写入数据的长度（两个字节为单位）。

UCHAR* uWriteData：需要写入的数据。

UCHAR* uUii：返回值，写入的标签的 UII。

UCHAR* uLenUii ：返回值，UII 长度。

UCHAR* uErrorCode：标签返回的 Error Code。只在函数返回失败，且 uErrorCode 不等于 0xFF 时有效。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.17 RmuEraseData ()

4.17.1 功能简介

该函数擦除标签数据。

4.17.2 函数原型

int WINAPI RmuEraseData (HANDLE hCom, UCHAR* uAccessPwd, UCHAR uBank, UCHAR* uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uErrorCode, UCHAR flagCrc);

4.17.3 返回值

1：成功擦除标签数据。

其他：擦除标签数据失败。

4.17.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uAccessPwd：标签的 ACCESS PASSWORD。

UCHAR uBank：标签的存储空间类型。

UCHAR* uPtr：起始地址的偏移量。

UCHAR uCnt：需要擦除的数据长度（两个字节为单位）。

UCHAR* uUii：标签的 UII。

UCHAR* uErrorCode：标签返回的 Error Code。只在函数返回失败，且 uErrorCode 不等于 0xFF 时有效。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.18 RmuLockMem ()

4.18.1 功能简介

该函数锁定标签的指定数据段。

4.18.2 函数原型

int WINAPI RmuLockMem (HANDLE hCom, UCHAR* uAccessPwd, UCHAR* uLockData, UCHAR* uUii, UCHAR* uErrorCode, UCHAR flagCrc);

4.18.3 返回值

1：成功锁定标签的指定数据段。

其他：锁定标签的指定数据段失败。

4.18.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uAccessPwd：标签的 ACCESS PASSWORD。

UCHAR* uLockData：命令的 LOCKDATA 数据段（3 字节）。

UCHAR* uUii：标签的 UII。

UCHAR* uErrorCode：标签返回的 Error Code。只在函数返回失败，且 uErrorCode 不等于 0xFF 时有效。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.19 RmuKillTag ()

4.19.1 功能简介

该函数销毁指定标签。

4.19.2 函数原型

int WINAPI RmuKillTag (HANDLE hCom, UCHAR* uKillPwd, UCHAR* uUii, UCHAR* uErrorCode, UCHAR flagCrc);

4.19.3 返回值

1：成功销毁指定标签。

其他：销毁指定标签失败。

4.19.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uKillPwd：标签的 Kill Password（32 位）。

UCHAR* uUii：标签的 UII。

UCHAR* uErrorCode：标签返回的 Error Code。只在函数返回失败，且 uErrorCode 不等于 0xFF 时有效。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.20 RmuGetVersion ()

4.20.1 功能简介

该函数读取 RMU 的硬件序列号和软件版本号。

4.20.2 函数原型

int WINAPI RmuGetVersion (HANDLE hCom, UCHAR* uSerial, UCHAR* uVersion, UCHAR flagCrc);

4.20.3 返回值

1：成功读取 RMU 的硬件序列号和软件版本号。

其他：读取 RMU 的硬件序列号和软件版本号失败。

4.20.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uSerial：RMU 的硬件序列号（6 个字节）。

UCHAR* uVersion：RMU 的软件版本号（1 个字节）。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.21 RmuGetAntenna ()

4.21.1 功能简介

该函数读取 RMU 的天线设置。

4.21.2 函数原型

int WINAPI RmuGetAntenna (HANDLE hCom, UCHAR* uAnt, UCHAR flagCrc);

4.21.3 返回值

1：成功读取 RMU 的天线设置。

其他：读取 RMU 的天线设置失败。

4.21.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uAnt：RMU 返回的 ANT 字节。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

4.22 RmuSetAntenna()

4.22.1 功能简介

该函数设置 RMU 的天线。

4.22.2 函数原型

int WINAPI RmuSetAntenna (HANDLE hCom, UCHAR uAnt, UCHAR flagCrc);

4.22.3 返回值

1：成功设置 RMU 的天线。

其他：设置 RMU 的天线失败。

4.22.4 输入参数

HANDLE hCom：通信端口句柄。

UCHAR* uAnt：RMU 返回的 ANT 字节。

UCHAR flagCrc：是否使用 CRC16 验证功能。1：使用 CRC 功能；0：不使用 CRC 功能。

附录 A：UII 格式

本文档中所谓的 UII 包含 PC bits。UII 的前两个字节是 PC（Protocol-control）位，其格式见表 A-1。

表 A-1 PC bits 格式

Bits 0 ~ 4	Bits 5 ~ 6	Bits 7 ~ 15
以 word（两个字节）为单位的 PC 和 UII 的总体长度	未定义	NSI（未使用）

注：UII 从低位开始传输。

PC 的前五位表示 PC 和 UII 的总体长度。例如，

PC bits 0~4（bin）	PC+UII 长度（字节）
00000	2
00001	4
00010	6
…	…

附录 B：Extensible bit Vectors (EBV)

EBV（Extensible Bit Vector）是一种能表示可延伸数据的数据结构。本文档中提到的 EBV 是以字节为单位的数组，数组中每个字节的最高位是延伸位。如果延伸位为 0，则表示该字节是最后一个字节；如果延伸位为 1，则表示后续还有有效字节。EBV 格式数据串表示的有效数据是从左到右忽略延伸位的比特流。

RMU 只支持一个字节和两个字节的 EBV 数据，其格式如下：

0	x x x x x x x		
1	x x x x x x x	0	x x x x x x x

其中每个字节的最高位是延伸位。当 EBV 需要表示的数小于等于 127 时可用一个字节，而当 EBV 需要表示的数大于 127 小于 16384 时需用两个字节。例如，

12： 0 0 0 0 1 1 0 0 ，

130：1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0。

附录 C：Error codes

对标签进行数据操作（读取标签数据、写入标签数据、擦除标签数据、锁定标签、销毁标签）时，如果标签遇到错误则会返回错误码（Error Code）。

表 C-1 标签错误码

Error Code 支持	Error Code 值（bin）	Error Code 名	Error 描述
Error-specific	00000000	其他错误	其他错误码未定义的错误
	00000011	存储空间溢出或未支持的 PC 值	指定的存储空间不存在或标签不支持指定的 PC 值
	00000100	存储空间被锁定	指定存储空间被锁定，不能进行读/写操作
	00001011	电力不足	因电力不足不能进行写入操作
Non-specific	00001111	不明错误	标签不支持 Error-specific 码

附录 D：Lock-Command Payload

Lock-Command Payload 是二十位的数据，高十位是 Mask，低十位是 Action。其格式见表 D-1。当 Mask 置为 1 时对应的 Action 位有效。Action 位的含义见表 D-2。

表 D-1 Lock-Command Payload 数据格式

Kill password		**Access password**		**UII memory**		**TID memory**		**User memory**	
19	18	17	16	15	14	13	12	11	10
Skip/ Write	Skip/ Write	Skip/ Write	Skip/ Write	Skip/ Write	Skip/ Write	Skip/ Write	Skip/ Write	Skip/ Write	Skip/ Write
9	8	7	6	5	4	3	2	1	0
Pwd read/ write	Perma lock	Pwd read/ write	Perma lock	Pwd write	Perma lock	Pwd write	Perma lock	Pwd write	Perma lock

表 D-2 Lock Action 位

Pwd-write	**Permalock**	**描述**
0	0	相应数据段在 OPEN 或 SECURED 状态下可写入
0	1	相应数据段在 OPEN 或 SECURED 状态下永久可写入，相应数据段不可锁定
1	0	相应数据段在 SECURED 状态下可写入，OPEN 状态下不可写入
1	1	相应数据段在任何状态下不可写入
Pwd-read/write	**Permalock**	**描述**
0	0	相应数据段在 OPEN 或 SECURED 状态下可读取和写入
0	1	相应数据段在 OPEN 或 SECURED 状态下永久可读取和写入，相应数据段不可锁定
1	0	相应数据段在 SECURED 状态下可读取和写入，OPEN 状态下不可读取和写入
1	1	相应数据段在任何状态下不可读取和写入

附录 E：插入字节实例

在实际通信过程中，当发送数据包的 SOF 和 EOF 之间出现 0xAA、0x55、0xFF 字节时，发送方应在该字节前插入一个 0xFF 字节。接收方接收到包含插入字节的数据后应删除插入字节并提取有效数据。插入字节不计入 LENGTH。例如，

需要发送的数据包（hex）：AA 04 55 00 01 55

实际发送的数据包（hex）：AA 04 FF 55 00 01 55

需要发送的数据包（hex）：AA 05 00 00 01 AA 55

实际发送的数据包（hex）：AA 05 00 00 01 FF AA 55

需要发送的数据包（hex）：AA 06 00 00 01 AA FF 55

实际发送的数据包（hex）：AA 06 00 00 01 FF AA FF FF 55

附录 F：Chinese 标准 && ETSI 标准

Chinese Standard：

1. Power && Frequency Table:

No.	Frequency rangs	Power level(e.r.p)	Power value
1	920,5-924,5	+33dbm	2W
2	840,5-844,5	+33dbm	2W
3	920,0-925,0	+20dbm	100mW
4	840,0-845,0	+20dbm	100mW

2. Power && Frequency Figure:

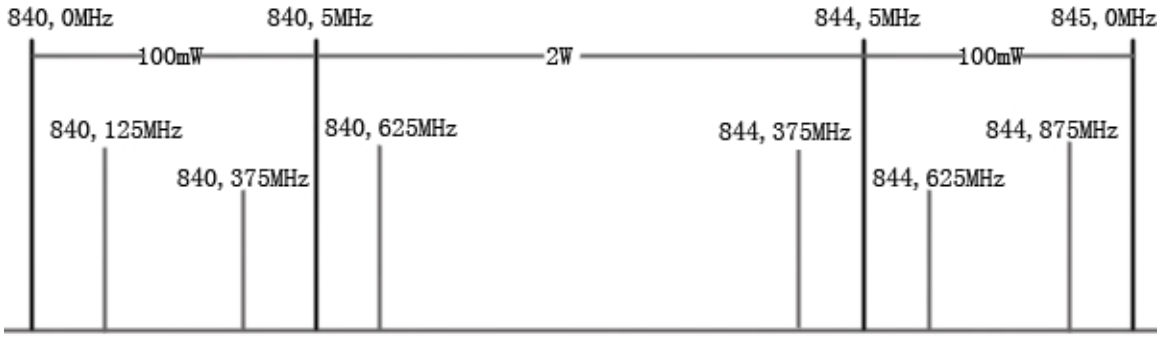

Figure 1: Frequency rangs of 840-845

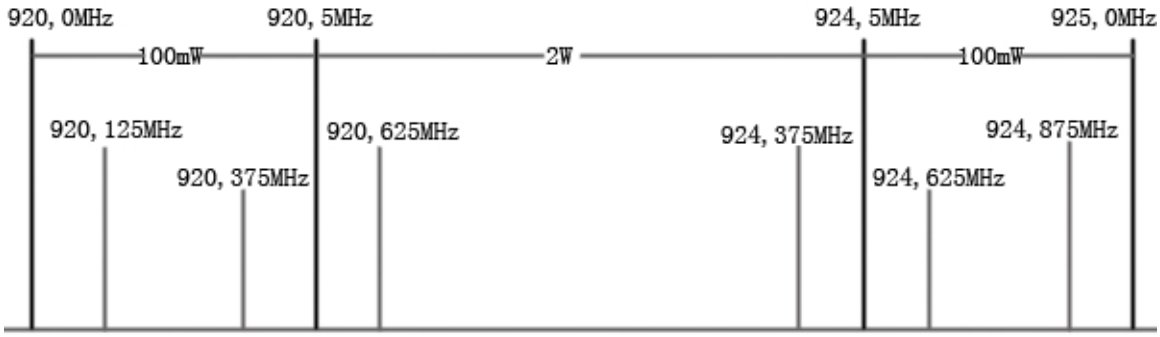

Figure 2: Frequency rangs of 920-925

3. Sub-banks: 250KHz

ETSI Standard：

1. Power && Frequency Table:

No.	Frequency rangs	Power level(e.r.p)	Power value
1	865,0-868,0	+20dbm	100mW
2	865,6-868,0	+27dbm	500mW
3	865,6-867,6	+33dbm	2W

2. Power && Frequency Figure:

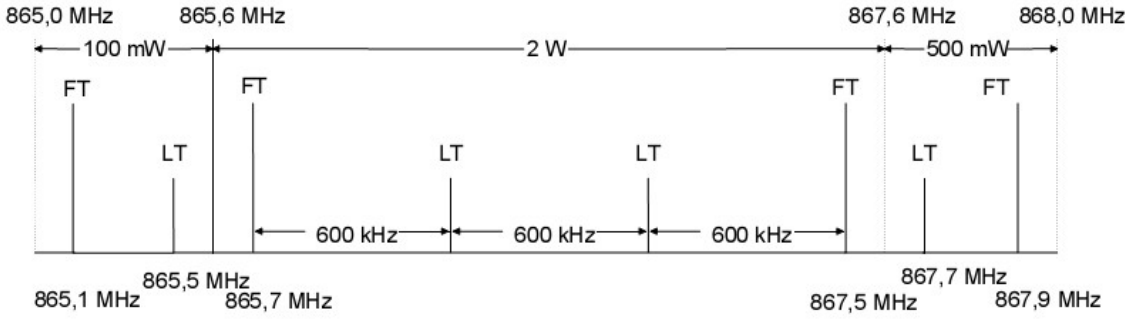

Figure 3: Frequency rangs of 865-868

3. Sub-banks: 200KHz, 600KHz