RhombusMF R/W 底层串行接口协议

1、 串行接口规格

RhombusMF R/W 通过串行接口与主机（单片机，微处理器，控制器等）实现数据通讯，按主机的命令要求完成相应操作。串行接口的数据帧为 1 个起始位，8 个数据位，无奇偶校验位和 1 个停止位，波特率 9600。在串行通讯过程中，最低有效字节最先传输，每个字节的最低有效位最先传输。

2、 控制字符

定义了四种控制字符，用以表示串行通信的开始、结束、应答及无应答，具体如下表：

定义	符号	值
开始符	STX	0x02
结束符	ETX	0x03
应答	ACK	0x06
无应答	NAK	0x15

3、 协议描述

通讯过程必须先由主机发送命令和数据给 RhombusMF R/W 读写引擎，然后 RhombusMF R/W 将命令执行结果状态和数据返回主机。需要通讯的双方首先通过握手过程启动一次通讯，然后进行数据的传递。

主机的命令发送过程如下表：

主机	数据传递方向	RhombusMF R/W	说明
STX	→		主机在20ms内未收到ACK或收到NAK，可重发STX；主机收到ACK后必须在45ms内开始命令数据块的发送，两个相邻字节发送的时间间隔必须小于15ms。在命令数据块的发送过程中，主机如果收到任何RhombusMF R/W发送信息，均表明主机与RhombusMF R/W通讯同步失步，主机可以停止发送命令，等待15ms后重新启动握手过程。
	←	ACK	
命令数据块 ＋ ETX	→		

首先主机发送 STX 并等待 RhombusMF R/W 返回 ACK 以启动数据块传递过程，若在 20ms 内未收到 ACK 或收到 NAK，主机可选择再次发送 STX 或停止通讯，进行通讯失败处理。主机收到 RHM8201ST 的 ACK 响应后，将包含操作命令、操作数据等的命令数据块发送至 RhombusMF R/W，最后发送一个 ETX 字符结束发送，然后等待其返回命令执行结果。

RhombusMF R/W 在收到主机命令后的 300ms 内完成命令执行并返回结果，命令执行结果的返回过程如下表：

RhombusMF R/W	数据传递方向	主机	说明
STX	→		RhombusMF R/W在 45ms 内未收到 ACK 将停止发送；RhombusMF R/W收到 ACK 后在 45ms 内开始响应数据块的发送。
	←	ACK	
响应数据块 ＋ ETX	→		

首先 RhombusMF R/W 发送 STX 并等待主机返回 ACK 以启动数据块传递过程，若在 45ms 内未收到 ACK，RhombusMF R/W 将停止本次通讯。RhombusMF R/W 收到主机的 ACK 响应后，将包含命令执行状态、执行结果数据等的响应数据块发送至主机，最后发送一个 ETX 字符结束发送。

4、 数据块的格式

A． 命令数据块

SeqNo	*Cmd*	*Len*	*Data[]*	*BCC*

SeqNo：1 个字节的数据包交换序列号，取值范围 0～255，经过一次正确的数据交换后，主机在发送下一个命令时，将序号加 1。RhombusMF R/W 将在响应数据块中返回最近接收的数据包序号。主机程序可以利用该序号加强通讯过程的完整性，也可以忽略该序号的处理以简化通讯过程；

Cmd： 1 个字节的操作命令符，一共定义了 25 条命令；

Len： 1 个字节的命令操作数长度，取值范围 0～22，为 0 表示无操作数；

Data[]：*Len* 个字节的命令操作数，若 *Len*＝0，无此项；

BCC： 1 个字节的校验和，为数据块中全部数据（*BCC* 自身除外）逐字节的异或值。

BCC=*SeqNo*⊕*Cmd*⊕Len⊕Data[]

B． 响应数据块

SeqNo	*Status*	*Len*	*Data[]*	*BCC*

SeqNo：1 个字节的最近接收的数据包交换序列号；

Status：1 个字节的命令执行结果状态值，为 0 表示命令执行成功，其它值表示出错代码；

Len： 1个字节的响应数据长度，取值范围 0～16，如果命令执行出错，该值为 0；

Data[]：*Len* 个字节的响应数据，若 *Len*＝0，无此项；

BCC： 1 个字节的校验和，为数据块中全部数据（*BCC* 自身除外）逐字节的异或值。

BCC=*SeqNo*⊕*Cmd*⊕Len⊕Data[]

5、 操作命令汇总

命令		参数		说明
名称	数值	发送	接收	
Request	0x41	_Mode	_TagType	发送询问命令,检查有效范围内是否有卡存在
Anticoll	0x42	Reserved	_Snr	开始防冲突检测并返回一张卡的序列号
Anticoll2	0x71	_Encoll，Reserved	_Snr	可允许或禁止多张卡进入,返回一张卡号
Select	0x43	_Snr	_Size	选择卡,返回卡的容量
Authentication	0x44	_Mode，Secnr	--	开始验证操作
Authentication2	0x72	_Mode，_Secnr，_Keynr	--	可选择密钥区验证
AuthKey 0x73		_Mode，_Secnr，_Key,	--	直接密码验证
Halt	0x45	--	--	将卡置于挂起模式
Read	0x46	_Adr	_Data	从卡中相应的块中读出 16 字节数据
Write	0x47	_Adr，_Data	--	向卡中相应的块中写入 16 字节数据
Increment	0x48	_Adr，_Value	--	增加访问块的数据值,将数据值存在卡内部寄存器
Decrement	0x49	_Adr，_Value	--	减少访问块的数据值,将数据值存在卡内部寄存器
Restore	0x4A	_Adr	--	将访问块的数据值恢复到卡内部寄存器
Transfer	0x4B	_Adr	--	将卡内部寄存器的数据值传送到访问块
Value	0x70	_Mode，_Adr，_Value _Trans，_Adr	--	包含加、减、恢复和传送
LoadKey 0x4C		_Mode，_Secnr，_nkey	--	改变 EEPROM 密钥区中的密钥
Reset	0x4E	_Msec	--	按指定毫秒数关闭天线使卡复位
Set_LED_Bit	0x50	--	--	读卡器上的 LED 灯显示为红色
Clr_LED_Bit	0x51	--	--	读卡器上的 LED 灯显示为绿色
Config	0x52	--	--	复位且配置 RHMMF1RW
Check_Write	0x53	_Snr，_Authmode，_Adr，_Data	--	将传送的数据和上次所写的数据进行比较
Buzzer	0x60	_Frequence，_Opentm，_Closetm，_Repcnt	--	控制蜂鸣器驱动输出按指定的间隔时间和重复次数动作
Close	0x3F	--	--	置 RhombusMF R/W 为待机状态
Read_E2	0x61	_Adr，_Length	Data	读 RhombusMF R/W中EEPROM 的内容
Write_E2	0x62	_Adr，_Length，_Data	--	写 RhombusMF R/W中EEPROM 的内容

6、 状态值列表

名称	值	描述
MI_OK，COMM_OK	0	函数调用成功
MI_NOTAGERR	1	有效区内没有卡
MI_CRCERR	2	CRC 错
MI_EMPTY	3	值溢出
MI_AUTHERR	4	不能验证
MI_PARITYERR	5	奇偶校验出错

MI_CODEERR	6	BCC 错误
MI_SENDERR	8	卡序列号出错
MI_KEYERR	9	密码错
MI_NOTAUTHERR	10	卡没有验证
MI_BITCOUNTERR	11	从卡中收到错误的数量位
MI_BYTECOUNTERR	12	从卡中得到错误数量的字节
MI_TRANSERR	14	TANSNFER 出错
MI_WRITEERR	15	WRITE 出错
MI_INCRERR	16	INCREMENT 错
MI_DECRERR	17	DECREMENT 错
MI_READERR	18	READ 错
MI_COLLERR	24	冲突错
MI_ACCESSTIMEOUT	27	访问超时
COMM_ERR	255	串行通信错误

7、 操作命令的详细描述

7.1 请求——Request

发送 Request 命令，检查在有效范围内是否有卡的存在。在选择一个新的卡进行操作之前必须执行此命令。

主机→**RhombusMF R/W**		
命令符	0x41	
长度	1	
Data[0] _Mode		_Mode=0，请求天线范围内处于 IDLE 状态的卡（处于 HALT 状态的除外） _Mode=1，请求天线范围内所有的卡

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT，MI_NOTAGERR，MI_BITCOUNTERR，COMM_ERR	
长度	2	
Data[0]	_tagtype(低字节)	当发生错误时，不返回任何内容
Data[1]	_tagtype(高字节)	当发生错误时，不返回任何内容

7.2 防碰撞——Anticoll

防冲突操作，必须在 Request 命令后立即调用。当知道了所要选择卡的序列号后，就没有必要调用，在 Request 命令后直接调用 Select 即可。

主机→**RhombusMF R/W**		
命令符	0x42	
长度	1	
Data[0] Reserved		保留字节，置为 0

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT，MI_NOTAGERR，MI_BITCOUNTERR，COMM_ERR	
长度	4	
Data[0]	snr(LL)	返回的卡的四位序列号
Data[1]	snr(LH)	
Data[2]	snr(HL)	
Data[3]	snr(HH)	

7.3 选择——Select

选择指定卡号的卡，返回卡的容量标志给主机

主机→**RhombusMF R/W**		
命令符：	0x43	
长度	4	
Data[0]	snr(LL)	四位卡的序列号
Data[1]	snr(LH)	
Data[2]	snr(HL)	
Data[3]	snr(HH)	

RhombusMF R/W→主机		
状态值	MI_OK,MI_QUIT,MI_NOTAGERR，MI_CRCERR，MI_PARITYERR，MI_BYTECOUNTERR，COMM_ERR	
长度	1	
Data[0]	_Size	卡的容量标志

7.4 证实——Authentication

对卡进行读、写、加、减等操作前，必须对卡进行证实操作。若卡中一扇区的密钥与 RHMMF1RW 中相应密钥存储区的密码匹配，则证实成功。

主机→**RhombusMF R/W**		
命令符	0x44	
长度	2	
Data[0]	_Mode	_Mode=0,利用密钥 A 进行验证 _Mode=1,利用密钥 B 进行验证
Data[1] _	SecNr	所访问卡的扇区号，必须小于 16，密钥区号与卡的扇区号相同

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT，MI_NOTAGERR，AUTHERR，MI_BITCOUNTERR，MI_PARITYERR，COMM_ERR	
长度	0	

7.5 暂停——**Halt**

将所有选择卡置为挂起状态。若要重新选择，则应用 ALL 模式调用 Request 。或将卡复位（如将卡离开天线操作区后再进入）或执行 Reset 。

主机→**RhombusMF R/W**		
命令符	0x45	
长度	0	

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT，COMM_ERR	
长度	0	

7.6 读——**Read**

此命令在所选的卡通过验证后，读取指定块的 16 个字节

主机→**RhombusMF R/W**		
命令符	0x46	
长度	1	
Data[0]	_Adr	所读数据的块地址

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT ， MI_NOTAGERR，AUTHERR ，MI_CRCERR ，MI_NOTAUTHERR ，MI_BITCOUNTERR，MI_PARITYERR，COMM_ERR	
长度	16	
Data[0]	Data（0）	读出的 16 字节的数据
Data[1]	Data（1）	
. . .	. . .	
Data[15]	Data（15）	

7.7 写——**Write**

此命令在所选的卡通过验证后，写入指定块的 16 字节。

主机→**RhombusMF R/W**		
命令符	0x47	
长度	17	
Data[0]	_Address	所读数据的块地址
Data[1]		要写入块的第一个字节
…		…
Data[16]		要写入块的最后一个字节

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT，MI_NOTAGERR，AUTHERR，MI_BITCOUNTERR，MI_WRITEERR，COMM_ERR	
长度	0	

7.8 加——**Increment**

读被访问的值块，检查数据的结构，用传输的值加值块的值，并将结果存在卡的内部寄存器中，值块有标准的格式。

主机→**RhombusMF R/W**		
命令符	0x48	
长度	5	
Data[0]	_Adr	要操作的数据块的地址
Data[1]	_value(ll)	要增加的值
Data[2]	_value(lh)	
Data[3]	_value(hl)	
Data[4]	_value(hh)	

RhombusMF R/W→主机		
状态值	MI_OK ， MI_QUIT ， MI_NOTAGERR ， MI_NOTAUTHERR ， MI_BITCOUNTERR ，MI_PARITYERR，COMM_ERR	
长度	0	

7.9 减——**Decrement**

读被访问的值块，检查数据的结构，用块的值减去传输的值，并将结果存在卡的内部寄存器中，值块有标准的格式。

主机→**RhombusMF R/W**		
命令符	0x49	
长度	5	
Data[0]	_Adr	要操作的数据块的地址
Data[1]	_value(ll)	要减的值
Data[2]	_value(lh)	
Data[3]	_value(hl)	
Data[4]	_value(hh)	

RhombusMF R/W→主机		
状态值	MI_OK ， MI_QUIT ， MI_NOTAGERR ， MI_NOTAUTHERR ， MI_BITCOUNTERR ，MI_PARITYERR，COMM_ERR	
长度	0	

7.10 恢复——**Restore**

读被访问的值块，检查数据的结构，并将结果存在卡的内部寄存器中，值块有标准的格式。

主机→**RhombusMF R/W**		
命令符	0x4a	
长度	1	
Data[0]	_Adr	要恢复的数据块的地址

RhombusMF R/W→主机		
状态值	MI_OK ， MI_QUIT ， MI_NOTAGERR ， MI_NOTAUTHERR ， MI_BITCOUNTERR ， MI_PARITYERR，COMM_ERR	
长度	0	

7.11 传送——**Transfer**

该命令将卡的内部寄存器内容传送给所选块。在此操作前必须通过验证，这个函数只能在 Incrment、Decrement,或 Restore 操作后操作。

主机→**RhombusMF R/W**		
命令符	0x4b	
长度	1	
Data[0] _Adr		卡中欲传送块的地址

RhombusMF R/W→主机		
状态值	MI_OK ， MI_QUIT ， MI_NOTAGERR ， MI_NOTAUTHERR ， MI_BITCOUNTERR ， MI_PARITYERR，COMM_ERR	
长度	0	

7.12 装载密钥——**Load_Key**

将一个新的密钥写入 EEPROM 存储器的密钥区中。执行完这个命令，要将 RhombusMF R/W 复位（如调用 Config 命令或 Close+config 命令）后才能将密码真正的写入。

主机→**RhombusMF R/W**		
命令符	0x4c	
长度	8	
Data[0]	_Mode	_Mode=0: 利用密码 A 进行验证 _Mode=1: 利用密码 B 进行验证
Data[1] _	Secnr	密钥所在的扇区号，必须小于 16
Data[2]	_nkey[0]	六字节密钥，最低字节先发送，最低字节的最低位先发送
…	…	
Data[7]	_ nkey[5]	

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT， MI_PARITYERR，COMM_ERR	
长度	0	

7.13 复位——**Reset**

射频电路关闭规定的时间，若_Msec=0,射频电路部分将一直处于关闭状态，直到下一个 Request 命令的到来。关闭射频能使天线内所有卡复位。

举例：

_Msec=0 : 射频电路一直关闭

_Msec=1 : 射频电路关闭 1ms

…

_Msec=0xff: 射频电路关闭 255ms

主机→**RhombusMF R/W**		
命令符	0x4e	
长度	1	
Data[0]	_Msec	射频电路关闭的时间（以毫秒为单位）

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT， COMM_ERR	
长度	0	

7.15 **LED** 灯点亮为红色——**Set_LED_Bit**

设置 **RhombusMF R/W** 的 LED 显示红色。

主机→**RhombusMF R/W**		
命令符	0x50	
长度	0	

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT， COMM_ERR	
长度	0	

7.16 **LED** 灯点亮为绿色——**Clr_LED_Bit**

设置 **RhombusMF R/W** 的 LED 显示绿色。

主机→**RhombusMF R/W**		
命令符	0x51	
长度	0	

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT， COMM_ERR	
长度	0	

7.17 配置——Config

RhombusMF R/W 每次上电复位之后，都必须先调用此命令对其进行初始化，才能进行进一步的操作。

主机→**RhombusMF R/W**		
命令符	0x52	
长度	0	

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT， COMM_ERR	
长度	0	

7.18 检查写——Check_Write

对写入卡的数据进行检查。将重新进行 Request/Select/Authenticated 操作，并将相应地址的数据与所给出的数据进行比较，正确返回 MIS_CHECK_OK，如果数据不相符，返回 MIS_CHK_COMPERR，发生其他错误返回 MIS_CHK_FAILED。

此过程中采用的证实密钥所在的密钥区区号与块_Adr 所在的扇区号相同。

主机→**RhombusMF R/W**		
命令符	0x53	
长度	22	
Data[0]	_Snr(ll)	所要检查的卡的序列号
Data[1]	_Snr(lh)	
Data[2]	_Snr(hl)	
Data[3]	_Snr(hh)	
Data[4]	_Authode	上一次写命令时验证的模式
Data[5]	_Adr	所要检查的数据块的地址
Data[6]	Data(0)	所要检查的 16 字节的数据
：	：	
：	：	
Data[20]	Data(21)	

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT，MIS_CHECK_OK，MIS_CHK_FAILED，COMM_ERR，MIS_CHK_COMPERR	
长度	0	

7.19 输出蜂鸣器信号——**Buzzer**

输出驱动外接蜂鸣器，频率、持续时间、间隔时间和重复次数可以控制。

主机→**RhombusMF R/W**		
命令符	0x60	
长度	4	
Data[0]	_Frequence	输出方波的频率，保留值，固定为 0
Data[1]	_Opentm	方波输出持续时间，取值（0-255），15ms 的分辨率
Data[2]	_Closetm	间隙时间，取值（0-255），15ms 的分辨率
Data[3]	_Repcnt	重复的次数

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT， COMM_ERR	
长度	0	

7.20 读 **EEPROM**

将 RhombusMF R/W 内 EEPROM 的数据读出，必须小于 0x80。从 0x80 开始是密钥存放区，不能被读出。

主机→**RhombusMF R/W**		
命令符	0x61	
长度	2	
Data[0]	_Adr	被读的 EEPROM 的首地址，必须小于 0x80
Data[1]	_Length	被读出数据长度（一次读出的数据要小于 20 个字节）

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT， COMM_ERR	
长度	_Length	
Data[0]	Byte（0）	读出的数据
:	:	
:	:	
Data[_Length-1]	Byte（_Length-1）	

7.21 写 **EEPROM**

将数据写入 **RhombusMF R/W** 内 EEPROM 中，在 EEPROM 的 0x00-0x0F 为只读产品信息区，0x10-0x2f 为启动寄存器初始化文件区，不要改写，0x80-0x1ff 为只读密钥区，可用 Load_key 命令写入。

主机→**RhombusMF R/W**		
命令符	0x62	
长度	_Length +2	

Data[0]	_Adr	EEPROM 的写入首地址，取值范围（0x30-0x7e）
Data[1]	_Length	写入的数据长度（一次写入的数据必须小于 20 个字节）
Data[2]	Data(0)	要写入的数据
Data[length+1]	Data(Length)	

RHMMF1RW→主机		
状态值	MI_OK，MI_QUIT， COMM_ERR	
长度	0	

7.22 值操作

对卡内的某一块进行加、减或数据备份，该块必须为值块格式，并支持自动传送，该块的地址与传输块的地址必须是在同一个扇区。

主机→**RhombusMF R/W**		
命令符	0x70	
长度	7	
Data[0]	_Mode	_Mode=0xc0：减 _Mode=0xc1：加 _Mode=0xc2：恢复
Data[1]	_Adr	卡内块地址，对该块进行值操作，取值范围 0-63
Data[2]	_value(ll)	当进行加或减操作时，为加数或减数；当进行恢复操作时，该值为空值。
Data[3]	_value(lh)	
Data[4]	_value(hl)	
Data[5]	_value(hh)	
Data[6]	_Trans_Adr	传输块地址，取值范围 0-63

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT，COMM_ERR，MI_NOTAERR，MI_BITCOUNTERR，MI_TRANSERR	
长度	0	

7.23 关闭 RhombusMF R/W——Close

设置 **RhombusMF R/W** 为待机状态。若要重新启动，调用 Config 命令。

主机→**RhombusMF R/W**		
命令符	0x3f	
长度	0	

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT， COMM_ERR	
长度	0	

7.24 防冲突 2——Anticoll2

开始防冲突操作。必须在调用了 Request 命令后立即调用。若已知卡的序列号，可以不进行防冲突操作而直接调用 Select。

主机→**RhombusMF R/W**		
命令符	0x71	
长度	2	
Data[0]	_Encoll	若为 1，则使能多张卡进入；若为 0，则不允许多张卡进入，若多张卡进入则返回错误
Data[1]	Reserve	Reserve=0

RHMMF1RW→主机		
状态值	MI_OK，MI_QUIT，COMM_ERR，MI_NOTAGERR，MI_COLLERR，MI_BITCOUNTERR	
长度	4	
Data[0]	Snr(LL)	
Data[1]	Snr(LH)	
Data[2]	Snr(HL)	
Data[3]	Snr(HH)	

7.25 证实 2——Authentication2

在对卡进行读、写、加、减等操作前，必须对卡进行验证。如果密钥区的密钥与扇区的密钥一致，则证实成功。

主机→**RhombusMF R/W**		
命令符	0x72	
长度	3	
Data[0]	_Mode	_Mode=0: 利用密钥 A 进行验证 _Mode=1: 利用密钥 B 进行验证
Data[1]	_SecNr	所要访问卡的扇区号，必须小于 16
Data[2]	_KeyNr	用于证实的密钥区号，必须小于 16
RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT，MI_NOTAGERR，MI_PAROTERR，MI_PAROTUERR，COMM_ERR	
长度	0	

7.26 直接密码证实——AuthKey

在对卡进行读、写、加、减等操作前，必须对卡进行验证。若传输的密码与扇区的密钥一致，则证实成功。

主机→**RhombusMF R/W**		
命令符	0x73	
长度	8	

Data[0]	_Mode	_Mode=0: 利用密钥 A 进行验证 _Mode=1: 利用密钥 B 进行验证
Data[1] _	SecNr	所要访问卡的扇区号，必须小于 16
Data[2]	_key(0)	用于证实的密码
:	:	
:	:	
Data[7]	_key(5)	

RhombusMF R/W→主机		
状态值	MI_OK，MI_QUIT，MI_NOTAGERR，MI_PAROTERR，MI_PAROTUERR，COMM_ERR	
长度	0	