UP-RFID-RT 型综合教学平台

13.56MHz 模块串口协议

(14443A)

博创智联科技有限公司

2016-08-15

13.56MHz_14443A 串口通讯协议

一、 指令集合

这部分主要介绍 13.56MHz 模块支持的 14443 协议下的指令汇总。按照 13.56MHz 模块的工作端分成两个部分介绍，ISO 14443A 协议指令是 13.56MHz 模块和 14443A 标签之间的通信协议指令，串口协议指令是 13.56MHz 模块与串口终端（如 PC 端、ARM 端）之间的通信协议指令。14443A 国际标准协议是对通信规则的标准，没有对 PICC（非接触式耦合卡）和 PCD（耦合设备）之间通信规定特定的命令。下面 14443A 的命令是基于 14443A 协议并结合 PHILIPS 公司的芯片资料而提取出的命令。

1. ISO 14443A 协议指令

表 1.1 ISO 14443A 协议指令

指令标识	指令代码（Hex）	说明
PICC_REQIDL	26	寻天线区内未进入休眠状态
PICC_REQALL	52	寻天线区内全部卡
PICC_ANTICOLL1	93	防冲撞（一级串联）
PICC_ANTICOLL2	95	防冲撞（二级串联）
PICC_AUTHENT1A	60	验证 A 密钥
PICC_AUTHENT1B	61	验证 B 密钥
PICC_READ	30	读块
PICC_WRITE	A0	写块
PICC_DECREMENT	C0	扣款
PICC_INCREMENT	C1	充值
PICC_RESTORE	C2	调块数据到缓冲区
PICC_TRANSFER	B0	保存缓冲区中数据
PICC_HALT	50	休眠（暂停）
PICC_RESET	E0	复位

注：表 1.1 是读写器上 STC 芯片和 LCRC632 芯片之间的通信命令，而实际使用这些命令的是 PICC 和 PCD 之间的交互。这些通信命令与标签相关，该标签工作在 ISO 14443A 协议上，具体信息可参照标签资料（PHILIPS 的资料）。

2. 串口协议指令

表 1.2 串口协议指令

指令标识	指令代码（Hex）	功能
REQUEST_A	0201	寻卡 A 卡
ANTICOLL	0202	A 卡防冲突
SELECT	0203	锁定 A 卡
SLEEP	0204	休眠 A 卡
AUTHENTICATION	0207	A 卡密钥验证
M1READ	0208	读 M1 卡命令
M1WRITE	0209	写 M1 卡
M1INITVAL	020A	初始化 M1 卡
M1READVAL	020B	读取 M1 卡
M1DECREAMENT	020C	M1 卡扣款
M1INCREAMENT	020D	M1 卡充值
ANT_CONTROL	010C	天线控制
GET_HARDMODEL	0104	获取版本号
SET_BAUDRATE	0101	设置波特率

注:M1 卡也就是 A 型卡的一种，这里的 M1 卡指模拟的一卡通、钱包等功能的卡。

二、 指令说明

1. ISO 14443A 协议指令说明

ISO 14443A 协议使用的是国际标准，详细说明参照 HF13.56M/国际标准/ISO 14443A 部分的文档。

2. 串口协议指令说明

13.56MHz 模块作为其他主机（如 PC）的外设通过串口通信，串口默认波特率为 19200。此处详细说明 13.56MHz 模块 14443A 协议下的串口指令以及各个指令帧结构。模块接收到的命令帧结构大致分为 6 部分，

主要分为帧头、校验和、参数以及命令字，具体如表 2.1 所示。

表 2.1 命令帧数据结构

字段	值	说明
SOF	0xAABB	主机（如 PC）与 13.56MHz 模块通信的命令帧起始字
Lenth	0xXXXX	一帧命令的长度，从 DEV_ID（包含）开始到 FCS（包含）结束为止的总长度
Dev_ID	0x0000	设备编号，用于扩展使用
CMD	0xXXXX	命令字，如寻卡命令字为 0x0201
Status/Flag	0xXX	状态字节或/和标志字节，主要在响应帧中出现
VData	...	可变字节的负载，发送时可以携带参数，接收时为响应数据
FCS	0xXX	校验和，从 LENGTH（不包含）开始到 FCS（不包含）结束的所有字节的异或值

注：请求和响应帧结构一致，最后校验和是字节，VDATA 也是按照字节计算，其余均为字，低位在前，高位在后。

2.1 REQUEST_A 命令

REQUEST_A 命令字是 0x0201，当 13.56MHz 模块通过串口接收到 REQUEST_A 命令后执行 ISO 14443A 协议下的寻卡命令。13.56MHz 模块可以寻找到处在天线辐射范围内的标签。REQUEST_A 命令帧结构如表 2.2 所示。

表 2.2 REQUEST_A 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	VData	FCS
AA BB	05 00	00 00	01 02	XX	10

注：VData 为请求参数，0x52 为寻所有卡，0x26 为寻未休眠的卡

13.56MHz 模块接收到请求命令并寻卡后会将寻到的卡号和执行的结果状态结果反馈给请求方（如 PC），响应帧结构如表 2.3 所示。

表 2.3 REQUEST_A 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status	VData	FCS
AA BB	08 00	00 00	01 02	XX	XX XX	fcs

表 2.3 中，Status 字节表示该命令执行是否成功，如果成功为 0x00，否则返回错误码。VData 为 2 个字节，为卡片类型代码，类型代码和卡片类型如表 2.4 所示。

表 2.4　卡片类型

代码（hex）	4400	0400	0200	0800	0403	4403
卡片类型（Mifare_）	UltraLight	One（S50）	One（S70）	Pro	ProX	DESFire

在表 2.4 中可以看到，REQUEST_A 命令返回的是卡的类型，而不是卡号，命令示例：

操作	交互帧
Send	AA BB 06 00 00 00 01 02 52 51
Recv	AA BB 08 00 00 00 01 02 00 04 00 07

2.2 ANTICOLL 命令

ANTICOLL 命令字是 0x0202，当 13.56MHz 模块通过串口接收到 ANTICOLL 命令后执行 ISO 14443A 协议下的 A 卡防冲撞命令。13.56MHz 模块接收到 ANTICOLL 命令帧结构如表 2.5 所示。

表 2.5　ANTICOLL 命令请求帧数据

SOF	**Lenth**	**Dev_ID**	**CMD**	**VData**	**FCS**
AA　BB	06 00	00 00	02 02	04	04

13.56MHz 模块接收到请求命令并执行后会将执行结果状态反馈给请求方（如 PC），响应帧结构如表 2.6 所示。

表 2.6　ANTICOLL 命令响应帧数据

SOF	**Lenth**	**Dev_ID**	**CMD**	**Status**	VData	**FCS**
AA　BB	0A 00	00 00	02 10	XX	UID	fcs

A 卡防冲撞命令需紧接着寻卡命令操作，单独使用不能获取卡号（单独使用寻卡命令也不能获取卡号）。在实际使用中 REQUEST_A、ANTICOLL、SELECT 命令结合使用，REQUEST_A 获取卡的类型，紧接着发送防冲撞命令，读取卡号并选定该卡。ANTICOLL 命令示例：

操作	交互帧
Send	AA BB 06 00 00 00 02 02 04 04
Recv	AA BB 0A 00 00 00 02 02 00 30 8F 6C 54 87

2.3 SELECT

SELECT 命令字是 0x0203，当 13.56MHz 模块通过串口接收到 SELECT 命令后执行 ISO 14443A 协议下的选择命令。13.56MHz 模块接收到 SELECT 命令帧结构如表 2.7 所示。

表 2.7　SELECT 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	VData	FCS
AA　BB	09 00	00 00	03 02	UID	fcs

13.56MHz 模块接收到请求命令并执行后会将执行结果状态反馈给请求方（如 PC），响应帧结构如表 2.8 所示。

表 2.8　SELECT 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status	VData	FCS
AA　BB	07 00	00 00	03 02	XX	08	fcs

SELECT 命令示例：

操作	交互帧
Send	AA BB 09 00 00 00 03 02 30 8F 6C 54 86
Recv	AA BB 07 00 00 00 03 02 00 08 09

2.4 SLEEP 命令

SLEEP 命令字是 0x0204，当 13.56MHz 模块通过串口接收到 SLEEP 命令后执行 ISO 14443A 协议下的休眠命令。命令帧结构如表 2.9 所示。

表 2.9　SLEEP 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	FCS
AA　BB	05 00	00 00	04 02	06

响应帧结构如表 2.10 所示。

表 2.10　SLEEP 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status	FCS
AA　BB	06 00	00 00	04 02	XX	fcs

SLEEP 命令示例：

操作	交互帧
Send	AA BB 05 00 00 00 04 02 06
Recv	AA BB 06 00 00 00 04 02 00 09

2.5 AUTHENTICATION 命令

AUTHENTICATION 命令字是 0x0207，当 13.56MHz 模块通过串口接收到 AUTHENTICATION 命令进行密钥验证，命令帧结构如表 2.11 所示。

表 2.11 AUTHENTICATION 命令请求帧数据

SOF	**Lenth**	**Dev_ID**	**CMD**	**VData**			**FCS**
AA BB	0D 00	00 00	07 02	Mode	Addr	Key	fcs
				XX	Blockaddr	passwd	

在表 2.11 请求帧中，VData 分为三个部分，第一部分为验证模式（Mode），如果使用密钥 A，model = 0x60，如果使用密钥 B，model = 0x61；第二部分为某扇区的起始块地址（一字节）；第三部分是密钥（六字节）。对 14443A 标签读写、一卡通或者钱包功能的充值、扣款、查询余额等功能均需要密钥验证后才能再次操作。13.56MHz 模块执行请求命令后会将执行结果状态反馈给请求方（如 PC），响应帧结构如表 2.12 所示。

表 2.12 AUTHENTICATION 命令响应帧数据

SOF	**Lenth**	**Dev_ID**	**CMD**	**Status**	**FCS**
AA BB	06 00	00 00	07 02	XX	fcs

AUTHENTICATION 命令会在读写标签的时候会首先验证，当验证成功才能进行读写操作。AUTHENTICATION 命令示例：

操作	交互帧	
Send	AA BB 0D 00 00 00 07 02 60 00 FF FF FF FF FF FF 65	密钥 A，0 扇区 0 块
Recv	AA BB 06 00 00 00 07 02 00 05	

2.6 M1READ 命令

M1READ 命令字是 0x0208，用于读取标签某个扇区某个块的内存数据。命令帧结构如表 2.13 所示。

表 2.13　M1READ 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	VData	FCS
AA　BB	13 00	00 00	08 02	XX	fcs

在表 2.13 请求帧中，VData 部分为读取的块绝对地址，块绝对地址 = 扇区*4 + 该扇区的块地址。响应帧结构如表 2.14 所示。

表 2.14　M1READ 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status/Flag	VData	FCS
AA　BB	16 00	00 00	08 02	XX	Data（16bit）	fcs

注：每次读写的时候均需要进行密码验证，因此需要先执行 AUTHENTICATION 命令，再执行 M1READ 命令。

M1READ 命令响应帧的 VData 部分为读取到的数据，长度固定为 16 个字节（有标签分区属性决定），M1READ 命令示例：

操作	交互帧（读取扇区 0 块 0 的数据）
Send	AA BB 06 00 00 00 08 02 00 0A
Recv	AA BB 16 00 00 00 08 02 00 2B 00 00 00 D4 FF FF FF 2B 00 00 00 01 FE 01 FE 21

对于钱包数据在内部存储上有一定的规则，将 16 字节分成 4 部分，每部分四个字节，第一部分和第三部分一致，第二部分是第一部分按位取反，第四部分可再分成两个相同的部分，第一个字节是第二个字节按位取反。

2.7 M1WRITE 命令

M1WRITE 命令字是 0x0209，用于将数据写入到指定的扇区和块地址处。命令帧结构如表 2.15 所示。

表 2.15　M1WRITE 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	VData		FCS
AA　BB	0F 00	00 00	09 02	Addr	Data	fcs
				addr	New Data	

写入命令帧的 VData 部分分为两个部分，第一个部分为写入的地址（块的绝对地址），addr = 扇区号*4 + Block 地址；第二个部分为写入的数据。响应帧结构如表 2.16 所示。

表 2.16　M1WRITE 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status/Flag	FCS
AA　BB	06 00	00 00	09 02	XX	fcs

M1WRITE 命令示例：

操作	交互帧（0 扇区块 2 写入 16 字节 0）
Send	AA BB 16 00 00 00 09 02 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09
Recv	AA BB 06 00 00 00 09 02 00 0B

2.8 M1INITVAL 命令

M1INITVAL 命令字是 0x020A，用于模拟一卡通初始化操作。命令帧结构如表 2.17 所示。

表 2.17　M1INITVAL 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	VData		FCS
AA　BB	0A 00	00 00	0A 02	Addr	Data	fcs
				Block addr	Value	

在表 2.17 请求帧中，VData 分为两个部分，第一部分为块绝对地址（Addr），addr = 扇区号*4 + Block 地址；第二部分为初始化的值，该值为整数的十六进制表示。响应帧结构如表 2.18 所示。

表 2.18　M1INITVAL 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status/Flag	FCS
AA　BB	06 00	00 00	0A 02	XX	fcs

M1INITVAL 命令示例：

操作	交互帧（0 扇区 1 块初始化 23 元）
Send	AA BB 0A 00 00 00 0A 02 01 17 00 00 00 1E
Recv	AA BB 06 00 00 00 0A 02 00 08

2.9 M1INCREMENT 命令

M1INCREMENT 命令字是 0x020D，用于给模拟的一卡通或者钱包充值。命令帧结构如表 2.19 所示。

表 2.19　M1INCREMENT 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	VData		FCS
AA　BB	0A 00	00 00	0D 02	Addr	Data	fcs
				Block addr	Value	

在表 2.19 请求帧中，VData 分为两个部分，第一部分为块绝对地址（Addr），addr = 扇区号*4 + Block 地址；第二部分为充值的值，该值为整数的十六进制表示（四字节，低位在前，高位在后）。响应帧格式如表 2.20 所示。

表 2.20　M1INCREMENT 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status/Flag	FCS
AA　BB	06 00	00 00	0D 02	XX	fcs

M1INCREMENT 命令示例：

操作	交互帧（0 扇区 1 块充值 12 元）
Send	AA BB 0A 00 00 00 0D 02 01 0C 00 00 00 1E
Recv	AA BB 06 00 00 00 0D 02 00 0F

2.10 M1DECREMENT 命令

M1DECREMENT 命令字是 0x020C，模拟一卡通或者钱包扣款或者消费。命令帧结构如表 2.20 所示。

表 2.21　M1DECREMENT 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	VData		FCS
AA　BB	0A 00	00 00	0C 02	Addr	Data	fcs
				BlockAddr	DecData	

在表 2.21 请求帧中，VData 分为两部分，第一部分为块绝对地址（Addr），addr=扇区号*4+Block 地址；第二部分为扣款的值，该值为整数的十六进制表示（四字节，低位在前，高位在后）。响应帧格式如表 2.22。

表 2.22　M1DECREMENT 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status/Flag	FCS
AA　BB	**06 00**	**00 00**	**0C 02**	**XX**	**fcs**

M1DECREMENT 命令示例：

操作	交互帧（0 扇区 1 块扣款 1 元）
Send	AA BB 0A 00 00 00 0C 02 01 01 00 00 00 0E
Recv	AA BB 06 00 00 00 0C 02 00 0E

2.11 M1READVAL 命令

M1READVAL 命令字是 0x020B，用于查询模拟卡内的余额。命令帧结构如表 2.23 所示。

表 2.23 M1READVAL 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	VData	FCS
AA BB	0E 00	00 00	0B 02	BlockAddr	fcs

在表 2.23 请求帧中，VData 为读取的绝对块地址，addr = 扇区号*4 + Block 地址。响应帧结构如表 2.24 所示。

表 2.24 M1READVAL 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status	VData	FCS
AA BB	06 00	00 00	0B 02	XX	Data	fcs

表 2.24 中 Status 为余额查询成功状态位，如果失败则为错误码，VData 为余额值，十六进制的整数值。

M1READVAL 命令示例：

操作	交互帧（查询扇区 0 块 1 处的余额，查询值为 32 元）
Send	AA BB 06 00 00 00 0B 02 01 08
Recv	AA BB 0A 00 00 00 0B 02 00 20 00 00 00 29

2.12 ANT_CONTROL 命令

ANT_CONTROL 命令字是 0x010C，控制天线打开和关闭。命令帧结构如表 2.25 所示。

表 2.25 ANT_CONTROL 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	VData	FCS
AA BB	06 00	00 00	0C 01	XX	fcs

在表 2.25 请求帧中，VData 为打开天线或关闭天线的参数，00 为关闭，否则为打开天线响应帧结构如表 2.26 所示。

表 2.26　ANT_CONTROL 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status	FCS
AA　BB	06 00	00 00	0C 01	XX	fcs

ANT_CONTROL 命令示例：

操作	交互帧
Send	AA BB 06 00 00 00 0C 01 00 08
Recv	AA BB 06 00 00 00 0C 01 00 0D

2.13 GET_HARDMODEL 命令

GET_HARDMODEL 命令字是 0x0104，该命令是对模块操作，因此不涉及到 ISO 14443A 相关的协议，有无标签在天线识别范围内对该命令无影响，当 13.56MHz 模块通过串口接收到 GET_HARDMODEL 命令帧结构如表 2.27 所示。

表 2.27　GET_HARDMODEL 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	FCS
AA　BB	05 00	00 00	04 01	fcs

13.56MHz 模块获取固件版本信息后将版本信息作为负载反馈给请求端，响应帧结构如表 2.28 所示。

表 2.28　GET_HARDMODEL 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status/Flag	VData	FCS
AA　BB	12 00	00 00	04 01	XX	…	fcs

注：VData 部分为版本信息的字节数组，无需转换，直接显示即可。

GET_HARDMODEL 命令示例：

操作	交互帧
Send	AA BB 05 00 00 00 04 01 05
Recv	AA BB 12 00 00 00 04 01 00 53 4C 36 30 31 46 2D 30 35 31 32 00 40

2.14 SET_BAUDRATE 命令

SET_BAUDRATE 命令字是 0x0101，用于对模块操作，不涉及到 ISO 14443A 相关的协议，有无标签在天线识别范围内对该命令无影响。当模块通过串口接收到 SET_BAUDRATE 命令帧结构如表 2.29 所示。

表 2.29 SET_BAUDRATE 命令请求帧数据

SOF	Lenth	Dev_ID	CMD	VData	FCS
AA BB	06 00	00 00	01 01	XX	fcs

在表 2.32 请求帧中，在 VData 部分用一个字节代表一个特定的波特率，13.56MHz 模块支持的波特率如表 2.30 所示。

表 2.30 13.56MHz 模块支持的波特率

编码	0	1	2	3	4	5	6	7
波特率	4800	9600	14400	19200	28800	38400	57600	115200

13.56MHz 模块接收到请求帧，VData 字节为表 2.30 中某一个编码，根据编码将模块波特率设置成对应的波特率，在不断电的情况下一只工作在该波特率下。同时用修改之前的波特率将设置结果响应给请求端，响应帧格式如表 2.31 所示。

表 2.31 SET_BAUDRATE 命令响应帧数据

SOF	Lenth	Dev_ID	CMD	Status/Flag	FCS
AA BB	06 00	00 00	01 10	XX	fcs

注：波特率的设置同时会影响 15693 协议，在不断电的情况下 15693 和 14443A 必须始终采用一致的波特率进行工作。

SET_BAUDRATE 命令示例：

操作	交互帧（设置波特率为 19200）
Send	AA BB 06 00 00 00 01 01 03 03
Recv	AA BB 06 00 00 00 01 01 00 00

附录

FCS（Frame check sequence）：

在 14443A 协议下，每个命令帧最后会追加一个字节作为该帧的校验和，从 LENGTH（不包含）开始到 FCS（不包含）结束的所有字节的异或值。计算方法（C 语言实现）：

```
//校验值计算
uint8 RC632_UartCalcFCS( uint8 *msg_ptr, uint8 len )
{
  uint8 x;
  uint8 xorResult;
  xorResult = 0;
  for ( x = 0; x < len; x++, msg_ptr++ )
    xorResult = xorResult ^ *msg_ptr;
  return ( xorResult );
}
```

说明：第一个参数是要计算的字节数组的起始地址，第二个参数为要计算的长度。