**UP-RFID-RT型综合教学平台**

**13.56MHz模块串口协议**

**(15693)**

**博创智联科技有限公司**

**2016-08-10**

**13.56MHz\_15639串口通讯协议**

## **指令集合**

这部分主要介绍13.56MHz模块支持的15693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，ISO15693协议指令是13.56MHz模块和15693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。

## **ISO15693协议指令**

**表1.1 ISO15693协议指令**

|  |  |  |
| --- | --- | --- |
| **指令标识** | **指令代码（Hex）** | **类型** |
| 寻卡 | 01 | 强制的 |
| 静止 | 02 | 强制的 |
| RFU | 03~1F | 强制的 |
| 读单一块 | 20 | 可选的 |
| 写单一块 | 21 | 可选的 |
| 锁定块 | 22 | 可选的 |
| 读多重块 | 23 | 可选的 |
| 写多重块 | 24 | 可选的 |
| 选择 | 25 | 可选的 |
| 重启准备 | 26 | 可选的 |
| 写AFI | 27 | 可选的 |
| 锁定AFI | 28 | 可选的 |
| 写DSFID | 29 | 可选的 |
| 锁定DSFID | 2A | 可选的 |
| 获取系统信息 | 2B | 可选的 |
| 获得多重块安全状态 | 2C | 可选的 |
| RFU | 2D~9F | 可选的 |
| IC生产厂家确定 | A0~DF | 自定义的 |
| IC生产厂家确定 | E0~FF | 专用的 |

## **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** |
| INVENTORY16 | 1000 | 寻卡（可寻多张） |
| INVENTORY | 1001 | 寻卡（不建议直接使用） |
| STAY\_QUIET | 1002 | 设置静默态 |
| SELECT | 1003 | 选择 |
| RESET\_TO\_READY | 1004 | 重置到准备态 |
| READ\_SM | 1005 | 读单个或者多个数据块 |
| WRITE\_SM | 1006 | 写单个数据块 |
| LOCK\_BLOCK | 1007 | 锁定数据块 |
| WRITE\_AFI | 1008 | 写应用族标志 |
| LOCK\_AFI | 1009 | 锁定应用族标志 |
| WRITE\_DSFID | 100A | 写数据存储格式标志 |
| LOCK\_DSFID | 100B | 锁定数据存储格式标志 |
| GET\_SYSINFO | 100C | 获取系统信息 |
| GET\_MULTIBLOCK\_SECURITY | 100D | 获取多个块的安全状态 |
| GET\_HARDMODEL | 0104 | 获取版本号 |
| SET\_BAUDRATE | 0101 | 设置波特率 |

## **指令说明**

## **ISO 15693协议指令说明**

ISO 15693协议使用的是国际标准，详细说明参照HF13.56M/国际标准/ISO15693部分的文档。

## **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15693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0x1000 |
| 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也是按照字节算的，其余均为字，低位在前，高位在后。

1. **INVENTORY16命令**

INVENTORY16命令字是0x1000，当13.56MHz模块通过串口接收到INVENTORY16命令后执行ISO 15693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未休眠的标签。INVENTORY16命令帧结构如表2.2所示。

**表2.2 INVENTORY16命令请求帧数据**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **FCS** |
| AA BB | 05 00 | 00 00 | 00 10 | 10 |

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 INVENTORY16命令响应帧数据**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status/Flag** | **VData** | **FCS** |
| AA BB | 0F 00 | 00 00 | 00 10 | 00 15 | UID | fcs |

表2.3中，Lenth字段不一定是0x000F，如果有多张卡，Lenth的长度是7+8\*标签的数量，Status/Flag第一个字节为状态字，用于表明寻卡成功与否，第二个字节为数据存储格式标志，例如DSFID为15，则响应帧的Status/Flag字段第二个字节就是15，VData部分为卡号，8字节，如果有多张卡，VData会以8字节为单位向后扩展。后续的命令大部分对特定标签进行操作，因此首先需要寻卡，然后才能进一步操作。

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧 |
| Send | AA BB 05 00 00 00 00 10 10 |
| Recv | AA BB 0F 00 00 00 00 10 00 45 76 9D 97 29 00 01 04 E0 E5 |

1. **STAY\_QUIET命令**

STAY\_QUIET命令字是0x1002，当13.56MHz模块通过串口接收到STAY\_QUIET命令后执行ISO 15693协议下的静止命令，处于静默态的标签不处理任何寻卡请求命令。当模块断电重启和接收到SELECT、RESET\_TO\_READY命令时退出静默状态进入到对应的选择态或者准备态。13.56MHz模块接收到STAY\_QUIET命令帧结构如表2.4所示。

**表2.4** STAY\_QUIET**命令请求帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | **FCS** |
| AA BB | 0D 00 | 00 00 | 02 10 | 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2.5所示。

**表2.5** STAY\_QUIET**命令响应帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status** | **FCS** |
| AA BB | 06 00 | 00 00 | 02 10 | XX | fcs |

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0） |
| Send | AA BB 0D 00 00 00 02 10 76 9D 97 29 00 01 04 E0 A2 |
| Recv | AA BB 06 00 00 00 02 10 00 12 |

1. **SELECT**

SELECT命令字是0x1003，当13.56MHz模块通过串口接收到SELECT命令后执行ISO 15693协议下的选择命令，处于静默态或准备态的标签会进入到选择态。处于选择态的标签可以执行该标签支持的任何指令。13.56MHz模块接收到SELECT命令帧结构如表2.6所示。

**表2.6 SELECT命令请求帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | **FCS** |
| AA BB | 0D 00 | 00 00 | 03 10 | 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2.7所示。

**表2.7 SELECT命令响应帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status** | **FCS** |
| AA BB | 06 00 | 00 00 | 03 10 | XX | fcs |

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0） |
| Send | AA BB 0D 00 00 00 03 10 76 9D 97 29 00 01 04 E0 A3 |
| Recv | AA BB 06 00 00 00 03 10 00 13 |

1. **RESET\_TO\_READY命令**

RESET\_TO\_READY命令字是0x1004，当13.56MHz模块通过串口接收到RESET\_TO\_READY命令后执行ISO 15693协议下的重启到准备态命令。处于选择态或静默态的标签通过该命令可以进入到准备态。13.56MHz模块接收到RESET\_TO\_READY命令帧结构如表2.8所示。

**表2.8 RESET\_TO\_READY命令请求帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | **FCS** |
| AA BB | 0E 00 | 00 00 | 04 10 | 02 UID | fcs |

在表2.8中，VData中UID前面有一个标志字节02，用于表明是地址请求标志还是选择请求标志，这里固定为02（地址请求标志）。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2.9所示。

**表2.9 RESET\_TO\_READY命令响应帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status** | **FCS** |
| AA BB | 06 00 | 00 00 | 04 10 | XX | fcs |

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0） |
| Send | AA BB 0E 00 00 00 04 10 02 76 9D 97 29 00 01 04 E0 A6 |
| Recv | AA BB 06 00 00 00 04 10 00 14 |

1. **READ\_SM命令**

READ\_SM命令字是0x1005，当13.56MHz模块通过串口接收到READ\_SM命令后执行ISO 15693协议下的读取单个或多个块数据命令。读取单个或者多个数据块通过VData中的参数来表明是读取单个块还是多个块，13.56MHz模块接收到READ\_SM命令帧结构如表2.10所示。

**表2.10 READ\_SM命令请求帧数据**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | | | | **FCS** |
| AA BB | 0E 00 | 00 00 | 05 10 | Flag | UID | Addr | Nblock | fcs |
| 02 | Uid | BlockAddr | n |

在表2.10请求帧中，VData分为四个部分，第一部分为标志（Flag），固定为02；第二部分为UID，表明是对哪张卡进行读取操作；第三部分是要读取的块起始地址（Address），表明是从哪块开始读取；第四部分是读取块的数量，当n = 1时读单块，当n > 1是读多个块，从Addr给定的地址开始连续读取n块的数据。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2.11所示。

**表2.11 READ\_SM命令响应帧数据**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status** | **VData** | **FCS** |
| AA BB | 0A 00 | 00 00 | 05 10 | XX | Data | fcs |

读取单个或者多个数据块响应帧如表2.11所示，Status为读取单个或多个块的成功与否的状态，VData为该块开始、请求读取的n块数据。每个块4个字节，因此此处Data的长度是4的整数倍，数据从左向右按照块地址从低到高排列。

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0，读取块3的数据，响应帧数据内容12 23 56 78） |
| Send | AA BB 10 00 00 00 05 10 02 76 9D 97 29 00 01 04 E0 03 01 A5 |
| Recv | AA BB 0A 00 00 00 05 10 00 12 34 56 78 1D |

1. **WRITE\_SM命令**

WRITE\_SM命令字是0x1006，当13.56MHz模块通过串口接收到WRITE\_SM命令后执行ISO 15693协议下的写入单个块数据命令。13.56MHz模块接收到WRITE\_SM命令帧结构如表2.12所示。

**表2.12 WRITE\_SM命令请求帧数据**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | | | | **FCS** |
| AA BB | 13 00 | 00 00 | 06 10 | Flag | UID | Addr | Data | fcs |
| 02 | Uid | BlockAddr | Data |

在表2.12请求帧中，VData分为四个部分，第一部分为标志（Flag），固定为02；第二部分为UID，表明是对哪张卡进行写入操作；第三部分是要写入的地址（Address）；第四部分是写入的数据（Data），每个块长度为四个字节，因此Data也为四个字节。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2.13所示。

**表2.13 WRITE\_SM命令响应帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status/Flag** | **FCS** |
| AA BB | 06 00 | 00 00 | 06 10 | XX | fcs |

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0，写入块8的数据，数据内容12 23 56 78） |
| Send | AA BB 13 00 00 00 06 10 02 76 9D 97 29 00 01 04 E0 08 12 34 56 78 A4 |
| Recv | AA BB 06 00 00 00 06 10 00 16 |

1. **LOCK\_BLOCK命令**

LOCK\_BLOCK命令字是0x1007，当13.56MHz模块通过串口接收到LOCK\_BLOCK命令后执行ISO 15693协议下的锁定数据块命令。锁定数据块需谨慎操作，数据块锁定之后不能解锁，而一旦锁定后该数据块不能再次写入。锁定命令适合使用在发放标签时写入一些只读的特殊信息。13.56MHz模块接收到LOCK\_BLOCK命令帧结构如表2.14所示。

**表2.14 LOCK\_BLOCK命令请求帧数据**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | | | **FCS** |
| AA BB | 0F 00 | 00 00 | 07 10 | Flag | UID | Addr | fcs |
| 02 | Uid | BlockAddr |

在表2.14请求帧中，VData分为三个部分，第一部分为标志（Flag），固定为02；第二部分为UID，表明是对哪张卡进行锁定操作；第三部分是要锁定的地址（Address）。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2.15所示。

**表2.15 LOCK\_BLOCK命令响应帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status/Flag** | **FCS** |
| AA BB | 06 00 | 00 00 | 07 10 | XX | fcs |

表2.15中Status/Flag为状态标志，如果成功返回00，失败返回18，为写入失败。因为锁定数据块实际是写该地址的锁存位，因此当失败时错误还是写卡失败。

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0，锁定07地址） |
| Send | AA BB 0F 00 00 00 07 10 02 76 9D 97 29 00 01 04 E0 07 A2 |
| Recv | AA BB 06 00 00 00 07 10 00 17 |

1. **WRITE\_AFI命令**

WRITE\_AFI命令字是0x1008，当13.56MHz模块通过串口接收到WRITE\_AFI命令后执行ISO 15693协议下的写入AFI命令。写入AFI主要用于给标签设置类型标识。13.56MHz模块接收到WRITE\_AFI命令帧结构如表2.16所示。

**表2.16 WRITE\_AFI命令请求帧数据**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | | | **FCS** |
| AA BB | 0F 00 | 00 00 | 08 10 | Flag | UID | AFI | fcs |
| 02 | Uid | Afi |

在表2.16请求帧中，VData分为三个部分，第一部分为标志（Flag），固定为02；第二部分为UID，表明是对哪张卡进行写入操作；第三部分是要写入的标志（AFI）。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2.17所示。

**表2.17 WRITE\_AFI命令响应帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status/Flag** | **FCS** |
| AA BB | 06 00 | 00 00 | 08 10 | XX | fcs |

表2.17中Status/Flag为状态标志，如果成功返回00，失败返回18，错误码含义为写入失败。

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0，AFI为12） |
| Send | AA BB 0F 00 00 00 08 10 02 76 9D 97 29 00 01 04 E0 12 B8 |
| Recv | AA BB 06 00 00 00 08 10 00 18 |

1. **LOCK\_AFI命令**

LOCK\_AFI命令字是0x1009，当13.56MHz模块通过串口接收到LOCK \_AFI命令后执行ISO 15693协议下的锁定AFI命令。锁定AFI需谨慎操作，AFI锁定之后不能解锁，而一旦锁定后该卡的AFI不能再次写入。锁定AFI命令适合使用在发放标签时为特殊的人群或者物品设置标志。13.56MHz模块接收到LOCK\_AFI命令帧结构如表2.18所示。

**表2.18 LOCK\_AFI命令请求帧数据**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | | **FCS** |
| AA BB | 0E 00 | 00 00 | 09 10 | Flag | UID | fcs |
| 02 | Uid |

在表2.18请求帧中，VData分为两个部分，第一部分为标志（Flag），固定为02；第二部分为UID，表明是对哪张卡进行锁定操作。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2.19所示。

**表2.19 LOCK \_AFI命令响应帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status/Flag** | **FCS** |
| AA BB | 06 00 | 00 00 | 09 10 | XX | fcs |

表2.19中Status/Flag为状态标志，如果成功返回00，失败返回18，为写入失败。因为锁定AFI实际是写AFI的锁存位，因此当失败时错误还是写卡失败。

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0） |
| Send | AA BB 0E 00 00 00 09 10 02 76 9D 97 29 00 01 04 E0 AB |
| Recv | AA BB 06 00 00 00 09 10 00 18 |

1. **WRITE\_DSFID命令**

WRITE\_DSFID命令字是0x100A，当13.56MHz模块通过串口接收到WRITE\_DSFID命令后执行ISO 15693协议下的写入数据存储格式标志（DSFID）命令。13.56MHz模块接收到WRITE\_DSFID命令帧结构如表2.20所示。在表2.20请求帧中，VData分为三个部分，第一部分为标志（Flag），固定为02；第二部分为UID，表明是对哪张卡进行写入操作；第三部分是要写入的标志（DSFID）。DSFID位没有锁定时可以反复写入，已经锁定则不能再次写入。

**表2.20 WRITE\_DSFID命令请求帧数据**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | | | **FCS** |
| AA BB | 0F 00 | 00 00 | 0A 10 | Flag | UID | DSFID | fcs |
| 02 | Uid | Flag |

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2.21所示。

**表2.21 WRITE\_DSFID命令响应帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status/Flag** | **FCS** |
| AA BB | 06 00 | 00 00 | 0A 10 | XX | fcs |

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0，DSFID为45） |
| Send | AA BB 0F 00 00 00 0A 10 02 76 9D 97 29 00 01 04 E0 45 ED |
| Recv | AA BB 06 00 00 00 0A 10 00 1A |

1. **LOCK\_DSFID命令**

LOCK\_DSFID 命令字是0x100B，当13.56MHz模块通过串口接收到LOCK\_DSFID命令后执行ISO 15693协议下的锁定数据存储格式标志命令。DSFID没有锁定时可以随便写入，但锁定后将不能再次写入，因此锁定操作需谨慎。DSFID主要用于标识卡内数据的存储格式或者编码等（如DSFID = 21时采用大端模式，DSFID = 12时采用小端模式）。13.56MHz模块接收到LOCK\_DSFID命令帧结构如表2.22所示。

**表2.22 LOCK\_DSFID命令请求帧数据**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | | **FCS** |
| AA BB | 0E 00 | 00 00 | 0B 10 | Flag | UID | fcs |
| 02 | Uid |

在表2.22请求帧中，VData分为两个部分，第一部分为标志（Flag），固定为02；第二部分为UID，表明是对哪张卡进行锁定操作。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2.23所示。

**表2.23 LOCK\_DSFID命令响应帧数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status/Flag** | **FCS** |
| AA BB | 06 00 | 00 00 | 0B 10 | XX | fcs |

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0） |
| Send | AA BB 0E 00 00 00 0B 10 02 76 9D 97 29 00 01 04 E0 A9 |
| Recv | AA BB 06 00 00 00 0B 10 00 1B |

1. **GET\_SYSINFO命令**

GET\_SYSINFO 命令字是0x100C，当13.56MHz模块通过串口接收到GET\_SYSINFO命令后执行ISO 15693协议下的获取系统信息命令。13.56MHz模块接收到GET\_SYSINFO命令帧结构如表2.24所示。

**表2.24 GET\_SYSINFO命令请求帧数据**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | | **FCS** |
| AA BB | 0E 00 | 00 00 | 0C 10 | Flag | UID | fcs |
| 02 | Uid |

在表2.24请求帧中，VData分为两个部分，第一部分为标志（Flag），固定为02；第二部分为UID，表明是后去哪张卡的系统信息。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2.25所示。

**表2.25 GET\_SYSINFO命令响应帧数据**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status** | **VData** | | | | | | **FCS** |
| AA BB | 14 00 | 00 00 | 0C 10 | XX | Flag | UID | DSFID | AFI | VICC | IC | fcs |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 信息标志 | 卡号 | 数据存储格式标志 | 应用族标志 | 内存容量信息 | IC参考值 |

响应帧如表2.25所示，VData部分为获取系统信息后的响应负载，分为六个部分。信息标志表明DSFID、AFI、VICC等是否支持，数据存储格式标志、应用族标志分别代表标签类型和内部数据的存储格式，内存容量表明该标签有多少个块，每个块可以存储多少个字节，最后IC参考值为标签厂商给定。下面分别介绍信息标志和内存容量部分的含义。信息标志定义如表2.26所示。

**表2.26 信息标志定义**

| **Bit** | **标志名字** | **值** | **描述** |
| --- | --- | --- | --- |
| B0 | DSFID | 0 | 不支持 DSFID。DSFID 域不出现 |
| 1 | 支持 DSFID。DSFID 域出现 |
| B1 | AFI | 0 | 不支持 AFI。AFI 域不出现 |
| 1 | 支持 AFI。AFI 域出现 |
| B2 | VICC内存容量 | 0 | 不支持信息的 VICC 内存容量。内存容量域不出现 |
| 1 | 支持信息的 VICC 内存容量。内存容量域出现 |
| B3 | IC参考 | 0 | 不支持信息的 IC 参考。IC 参考域不出现 |
| 1 | 支持信息的 IC 参考。IC 参考域出现 |
| B4~B7 | RFU | 0 | 保留 |

内存容量信息如表2.27所示。

**表2.27 内存容量信息**

|  |  |  |
| --- | --- | --- |
| B15~B13 | B12~B8 | B8~B0 |
| RFU | 块容量 | 块数 |

注：块容量信息和块数均比实际的小1，例如获取到内存容量为031B，则实际上每块的容量是4，块的数目是1C，原因是块容量和块数均从0开始，例如块00 01 02 03，实际是块是4块。

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0，VData：0F 76 9D 97 29 00 01 04 E0 45 45 1B 03 01） |
| Send | AA BB 0E 00 00 00 0C 10 02 76 9D 97 29 00 01 04 E0 AE |
| Recv | AA BB 14 00 00 00 0C 10 00 0F 76 9D 97 29 00 01 04 E0 45 45 1B 03 01 BA |

1. **GET\_MULTIBLOCK\_SECURITY命令**

GET\_MULTIBLOCK\_SECURITY 命令字是0x100D，当13.56MHz模块通过串口接收到GET\_MULTIBLOCK\_SECURITY命令后执行ISO 15693协议下的获取多个块的安全状态命令。13.56MHz模块接收到GET\_MULTIBLOCK\_SECURITY命令帧结构如表2.28所示。

**表2.28 GET\_MULTIBLOCK\_SECURITY命令请求帧数据**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **VData** | | | | **FCS** |
| AA BB | 10 00 | 00 00 | 0D 10 | Flag | UID | StartAddr | EndAddr | fcs |
| 02 | Uid | Start | End |

在表2.28请求帧中，VData分为四个部分，第一部分为标志（Flag），固定为02；第二部分为UID，表明是需要获取哪张卡的相关信息；第三部分为获取块的起始地址；第四部分为获取块的结束地址。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2.29所示。

**表2.29 GET\_MULTIBLOCK\_SECURITY命令响应帧数据**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Dev\_ID** | **CMD** | **Status/Flag** | **VData** | **FCS** |
| AA BB | XX XX | 00 00 | 0D 10 | XX | XX … | fcs |

在响应帧中，各个块的安全状态由一个字节表示，如果锁定该字节为01，未锁定为00，VData的长度与获取的多个模块安全状态的起始和结束地址相关。

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（卡号为 76 9D 97 29 00 01 04 E0，获取00到08地址的安全状态信息） |
| Send | AA BB 10 00 00 00 0D 10 02 76 9D 97 29 00 01 04 E0 00 09 A6 |
| Recv | AA BB 0F 00 00 00 0D 10 00 01 00 00 01 01 00 00 01 00 1D |

1. **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 15693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2.30所示。

**表2.30 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2.31所示。

**表2.31 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 |

VData部分为版本信息的字节数组，无需转换，直接显示即可。操作示例如下：

|  |  |
| --- | --- |
| 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 |

1. **SET\_BAUDRATE 命令**

SET\_BAUDRATE命令字是0x0101，该命令是对模块操作，因此不涉及到ISO 15693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SET\_BAUDRATE命令帧结构如表2.32所示。

**表2.32 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2.33所示。

**表2.33 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2.33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改波特率下。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2.34所示。

**表2.34 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。

示例：

|  |  |
| --- | --- |
| 操作 | 交互帧（设置波特率为115200） |
| Send | AA BB 06 00 00 00 01 01 07 07 |
| Recv | AA BB 06 00 00 00 01 01 00 00 |

**附录**

FCS（Frame check sequence）：

在15693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。