**UP-RFID-RT型综合教学平台**

**原理机串口通讯协议**

**(15693)**

**博创智联科技有限公司**

**2016-08-10**

**13.56MHz\_15639原理机串口通讯协议**

## **指令集合**

这部分主要介绍13.56MHz原理机模块支持的15693协议下的指令汇总。按照13.56MHz原理机模块的工作端分成两个部分介绍，ISO15693协议指令是原理机模块和15693标签之间的通信协议指令，串口协议指令是原理机模块与串口终端（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** |
| I\_content\_request | 0xA0 | 获取版本信息（未实现） |
| I\_keep\_silence | 0x02 | 保持静默态 |
| I\_read\_single\_block | 0x20 | 读单个数据块 |
| I\_write\_single\_block | 0x21 | 写单个数据块 |
| I\_lock\_single\_block | 0x22 | 锁定单个数据块 |
| I\_read\_multiple\_block | 0x23 | 读多个数据块 |
| I\_write\_multiple\_block | 0x24 | 写多个数据块(暂不支持) |
| I\_vicc\_select | 0x25 | 选择 |
| I\_vicc\_reset | 0x26 | 复位到准备态 |
| I\_write\_AFI | 0x27 | 写AFI |
| I\_lock\_AFI | 0x28 | 锁定AFI |
| I\_write\_DSFID | 0x29 | 写DSFID |
| I\_lock\_DSFID | 0x2A | 锁定DSFID |
| I\_get\_information | 0x2B | 获取系统信息 |
| I\_get\_status | 0x2C | 获取多个块的安全状态 |
| I\_buzzer | 0x2D | 蜂鸣器 |

## **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原理机模块作为其他主机（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原理机模块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、帧尾、参数、校验和以及命令字，具体如表2.1所示。

**表2.1 命令帧数据结构**

|  |  |  |
| --- | --- | --- |
| **字段** | **值** | **说明** |
| SOF | 0x02 | 帧头，固定为0x02 |
| Lenth | 0xXX | 一帧命令的长度，从SOF（包含）开始到EOF（包含）结束为止的总长度 |
| Flag | 0xXX | 标志位，包含编码模式、数据率、载波信息，可参照表2.4 |
| CMD | 0xXX | 命令字节，如获取系统信息为0x2B，具体值参照表1.2 |
| VData | 0xXX | 可变长的数据值 |
| FCS | 0xXX | 校验和，从Lenth（包含）字节到FCS（不包含）开始字节的所有字节的和模256 |
| EOF | 0x03 | 帧尾，固定为0x03 |

13.56MHz原理机模块主要学习RFID的原理及分析PCD和PICC之间的通信波形，原理机模块对部分请求命令没有反馈信息，响应帧和请求帧结构不一致，响应帧结构如表2.2所示。

**表2.2 响应帧数据结构**

|  |  |  |
| --- | --- | --- |
| **字段** | **值** | **说明** |
| CMD | 0xXX | 命令字节，可参照表1.2 |
| Flag | 0xXX | 标志，标志请求帧是否正确执行，参照表2.3 |
| VData | 0xXX | 响应帧负载 |
| CRC16/CCITT | 0xXXXX | 16位的CRC校验值 |

响应帧标志字节和请求帧标志字节的各个位含义各不相同，响应帧的标志位的定义如表2.3所示，请求帧的标志位定义如表2.4所示。

**表2.3 响应标志1-8位的定义**

|  |  |  |  |
| --- | --- | --- | --- |
| **位(Bit)** | **标志名称** | **值** | **描述** |
| b1 | 出错标志 | 0 | 没有出错 |
| 1 | 检测到错误。错误码值为0F |
| b4 | 扩展协议 | 0 | 无协议格式扩展 |
| 1 | 协议格式扩展 |
| b5-b8，b2-b3 | RFU | 0 | 保留 |

**表2.4 请求标志字定义**

|  |  |  |  |
| --- | --- | --- | --- |
| **位(Bit)** | **标志名称** | **值** | **描述** |
| b0 | 位编码模式 | 0 | 使用一个副载波的位编码(ASK) |
| 1 | 使用两个副载波的位编码(FSK) |
| b1 | 数据速率 | 0 | 低数据速率 |
| 1 | 高数据速率 |
| b2 | 数据编码模式 | 0 | 4取1编码模式 |
| 1 | 256取1编码模式 |
| b3-b7 | 未使用 | 0 | 默认为0 |

1. **I\_get\_information命令**

I\_get\_infomation 命令字是0x2B，当原理机模块通过串口接收到I\_get\_infomation命令后执行ISO 15693协议下的获取系统信息命令。原理机模块接收到I\_get\_infomation命令帧结构如表2.5所示。

**表2.5 I\_get\_infomation命令请求帧格式**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **FCS** | **EOF** |
| 02 | 06 | XX | 2B | Fcs | 03 |

在表2.5请求帧中，Flag字节为标志，具体的值请参照表2.4。FCS为Length字节+Flag字节+CMD字节的和取低8位。该命令为对原理机操作的第一个命令，用于获取卡号信息，其他的操作大部分需要卡号信息。13.56MHz原理机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 I\_get\_infomation命令响应帧格式**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CMD** | **Flag** | **VData** | | | | | | **CRC16/CCITT** |
| 2B | XX | Flag | UID | DSFID | AFI | VICC | IC | Crc-16 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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2.6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2.7所示。

**表2.7 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2.8所示。

**表2.8 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，例如块号00 01 02 03，实际是块是4块。获取系统信息的示例如表2.9所示。

**表2.9 I\_get\_information示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **交互帧（hex）** | **说明** |
| Send | 02 06 02 2B 33 03 | 请求获取系统信息 |
| Recv | 2B 00 0F 76 9D 97 29 00 01 04 E0 45 45 1B 03 01 EC CD | 成功获取系统信息 |

表2.9为获取系统信息的命令示例，获取到卡号为E0 04 01 00 29 97 9D 76,AFI和DSFID均为45，每个块的容量是4= 03+1，一共有1C = 1B+1块，IC参考值为1，CRC-16/CCITT值为0xECCD。

1. **I\_keep\_silence命令**

I\_keep\_silence命令码是0x02，当原理机模块通过串口接收到I\_keep\_silence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I\_vicc\_select、I\_vicc\_reset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原理机模块接收到I\_keep\_silence命令帧结构如表2.10所示。

**表2.10 I\_keep\_silence命令请求帧格式**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | **FCS** | **EOF** |
| 02 | 0E | XX | 02 | UID | Fcs | 03 |

13.56MHz原理机模块接收到请求命令并执行保持静默命令，无响应帧反馈给请求方（如PC），请求帧示例如表2.11所示。

**表2.11 I\_keep\_silence示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 0E 02 02 76 9D 97 29 00 01 04 E0 CA 03 | 请求使标签E004010029979D76进入休眠态 |

1. **I\_vicc\_select 命令**

I\_vicc\_select命令码是0x25，当13.56MHz原理机模块通过串口接收到I\_vicc\_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原理机模块接收到I\_vicc\_select命令帧结构如表2.12所示。

**表2.12 I\_vicc\_select命令请求帧格式**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | **FCS** | **EOF** |
| 02 | 0E | XX | 25 | UID | Fcs | 03 |

13.56MHz原理机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 I\_vicc\_select命令响应帧格式**

|  |  |  |
| --- | --- | --- |
| **CMD** | **Flag** | **CRC16/CCITT** |
| 25 | XX | XX XX |

I\_vicc\_select命令示例如表2.14所示。

**表2.14 I\_vicc\_select示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **交互帧（hex）** | **说明** |
| Send | 02 0E 02 25 76 9D 97 29 00 01 04 E0 ED 03 | 请求标签进入选择态 |
| Recv | 25 00 78 F0 | 成功执行进入选择态 |

1. **I\_vicc\_reset 命令**

I\_vicc\_reset命令码是0x26，当13.56MHz原理机模块通过串口接收到I\_vicc\_reset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I\_vicc\_reset命令帧结构如表2.15所示。

**表2.15 I\_vicc\_reset命令请求帧格式**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | **FCS** | **EOF** |
| 02 | 0E | XX | 26 | UID | Fcs | 03 |

13.56MHz原理机模块接收到请求命令后处理请求并进入到准备状态，无响应信息反馈给请求方（如PC），请求帧示例如表2.16所示。

**表2.16 I\_vicc\_reset命令响应帧格式**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 0E 02 26 76 9D 97 29 00 01 04 E0 EE 03 | 请求使标签进入到准备态 |

1. **I\_read\_single\_block命令**

I\_read\_single\_block命令码是0x20，当13.56MHz原理机模块通过串口接收到I\_read\_single\_block命令后执行ISO 15693协议下的读取单个数据命令。读取单个数据块的具体地址包含在VData中，13.56MHz原理机模块接收到I\_read\_single\_block命令帧结构如表2.17所示。

**表2.17 I\_read\_single\_block命令请求帧格式**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | | **FCS** | **EOF** |
| 02 | 0F | XX | 20 | UID | Addr | fcs | 03 |

在表2.17请求帧中，VData分为两个部分，第一部分为标签ID（UID）；第二部分为读取的地址（Addr）。VData表明您是要读取哪张卡的什么位置的数据。13.56MHz原理机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2.18所示。

**表2.18 I\_read\_single\_block命令响应帧格式**

|  |  |  |  |
| --- | --- | --- | --- |
| **CMD** | **Flag** | **VData** | **CRC16/CCITT** |
| 20 | XX | Data（4Bytes） | XX XX |

读取单个数据块响应帧如表2.18所示， VData为读取到的实际数据字节序列。命令示例如表2.19所示。

**表2.19 I\_read\_single\_block示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **交互帧（hex）** | **说明** |
| Send | 02 0F 02 20 76 9D 97 29 00 01 04 E0 00 E9 03 | 请求读取标签00地址的数据 |
| Recv | 20 00 23 42 34 32 74 15 | 读取成功，数据为23 42 34 32 |

1. **I\_write\_single\_block命令**

I\_write\_single\_block命令码是0x21，当13.56MHz原理机模块通过串口接收到I\_write\_single\_block命令后执行ISO 15693协议下的写入单个数据块命令。13.56MHz原理机模块接收到I\_write\_single\_block命令帧结构如表2.20所示。

**表2.20 I\_write\_single\_block命令请求帧格式**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | | | **FCS** | **EOF** |
| 02 | 13 | XX | 21 | UID | Addr | Data | fcs | 03 |

在表2.20请求帧中，VData分为三个部分，第一部分为UID，表明是对哪张卡进行写入操作；第二部分是要写入的地址（Address）；第三部分是写入的数据（Data），每个块长度为四个字节，因此Data也为四个字节。13.56MHz原理机模块接收到请求命令后将数据写入到对应的地址，无响应帧。写入单个数据块命令示例如表2.21所示。

**表2.21 I\_write\_single\_block命令示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 13 02 21 76 9D 97 29 00 01 04 E0 03 34 34 34 33 C0 03 | 请求将数据34 34 34 33写入到03地址 |

1. **I\_read\_multiple\_block命令**

I\_read\_multiple\_block命令码是0x23，当13.56MHz原理机模块通过串口接收到I\_read\_multiple\_block命令后执行ISO 15693协议下的读取多个数据块命令。读取多个数据块通过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I\_read\_multiple\_block命令帧结构如表2.22所示。

**表2.22 I\_read\_multiple\_block命令请求帧格式**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | | | **FCS** | **EOF** |
| 02 | 10 | XX | 23 | UID | Addr\_Start | Addr\_End | fcs | 03 |

在表2.22请求帧中，VData分为三个部分，第一部分为UID，表明是对哪张卡进行读取操作；第二部分是要读取的块起始地址（Address），表明是从哪块开始读取；第三部分是读取块的结束地址。13.56MHz原理机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2.23 I\_read\_multiple\_block命令响应帧格式**

|  |  |  |  |
| --- | --- | --- | --- |
| **CMD** | **Flag** | **VData** | **CRC16/CCITT** |
| 23 | XX | Data | XX XX |

读取多个数据块响应帧如图2.23所示，VData部分为读取到的内存数据字节序列。每个块内存容量是4个字节，因此VData的长度为结束地址减起始地址乘以四，数据从左向右按照块地址从低到高排列。读取多个块的命令示例如表2.24所示。

**表2.24 I\_read\_multiple\_block命令示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 10 02 23 76 9D 97 29 00 01 04 E0 00 07 F4 03 | 请求读取00到07地址的数据 |
| Recv | 23 00 23 42 34 32 87 65 43 21 12 34 56 78 12 34 56 78 55 55 55 55 12 12 12 12 12 34 56 78 12 32 13 23 66 06 | 地址0处的数据23 42 34 32  地址1处的数据87 65 43 21  … … … |

1. **I\_lock\_single\_block命令**

I\_lock\_single\_block命令码是0x22，当13.56MHz原理机模块通过串口接收到I\_lock\_single\_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原理机模块接收到I\_lock\_single\_block命令帧结构如表2.25所示。

**表2.25 I\_lock\_single\_block命令请求帧格式**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | | **FCS** | **EOF** |
| 02 | 0F | XX | 22 | UID | Addr | fcs | 03 |

在表2.25请求帧中，VData分为两个部分，第一部分为UID，表明是对哪张卡进行锁定操作；第二部分是要锁定的地址（Address）。13.56MHz原理机模块接收到请求命令执行锁定操作，无反馈信息给请求方（如PC），锁定示例如表2.26所示。

**表2.26 I\_lock\_single\_block命令示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 0F 02 22 76 9D 97 29 00 01 04 E0 00 EB 03 | 请求锁定00块 |

1. **I\_write\_AFI 命令**

I\_write\_AFI命令码是0x27，当13.56MHz原理机模块通过串口接收到I\_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原理机模块接收到I\_write\_AFI命令帧结构如表2.27所示。

**表2.27 I\_write\_AFI命令请求帧格式**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | | **FCS** | **EOF** |
| 02 | 0E | XX | 27 | UID | AFI | fcs | 03 |

在表2.27请求帧中，VData分为两个部分，第一部分为UID，表明是对哪张卡进行写入操作；第二部分是要写入的标志（AFI）。13.56MHz原理机模块接收到请求命令并写入AFI，原理机模块没有反馈给请求方（如PC）的信息，写入AFI命令示例如表2.28所示。

**表2.28 I\_write\_AFI命令示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 0F 02 27 76 9D 97 29 00 01 04 E0 34 24 03 | 请求写入AFI，AFI=34 |

1. **I\_lock\_AFI命令**

I\_ lock \_AFI命令码是0x28，当13.56MHz原理机模块通过串口接收到I\_ lock \_AFI命令后执行ISO 15693协议下的锁定AFI命令。锁定AFI主要用于给标签设置类型标识后不希望再次修改的情况。13.56MHz原理机模块接收到I\_ lock \_AFI命令帧结构如表2.29所示。

**表2.29 I\_lock\_AFI命令请求帧格式**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | **FCS** | **EOF** |
| 02 | 0E | XX | 28 | UID | fcs | 03 |

13.56MHz原理机模块接收到请求命令后执行锁定操作，无反馈信息给请求端（如PC），检查是否锁定可以修改AFI后再读取|AFI，如果不能再次修改则锁定成功。锁定AFI命令示例如表2.30所示。

**表2.30 I\_lock\_AFI命令示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 0E 02 28 76 9D 97 29 00 01 04 E0 F0 03 | 请求锁定AFI |

1. **I\_write\_DSFID命令**

I\_write\_DSFID命令码是0x29，当13.56MHz原理机模块通过串口接收到I\_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I\_write\_DSFID命令帧结构如表2.31所示。在表2.31请求帧中，VData分为两个部分，第一部分为UID，表明是对哪张卡进行写入操作；第二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2.31 I\_write\_DSFID命令请求帧格式**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | | **FCS** | **EOF** |
| 02 | 0F | XX | 29 | UID | DSFID | fcs | 03 |

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），写入DSFID请求示例如表2.32所示。

**表2.32 I\_write\_DSFID写入命令示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 0F 02 29 76 9D 97 29 00 01 04 E0 78 6A 03 | 请求写入DSFID |

1. **I\_lock\_DSFID命令**

I\_lock\_DSFID 命令码是0x2A，当13.56MHz原理机模块通过串口接收到I\_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原理机模块接收到I\_lock\_DSFID命令帧结构如表2.33所示。

**表2.33 I\_lock\_DSFID命令请求帧格式**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | **FCS** | **EOF** |
| 02 | 0E | XX | 2A | UID | fcs | 03 |

13.56MHz原理机模块接收到请求命令执行锁定DSFID操作，没有反馈信息给请求方（如PC），锁定DSFID请求示例如表2.34所示。

**表2.34 I\_lock\_DSFID命令示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 0E 02 2A 76 9D 97 29 00 01 04 E0 F2 03 | 请求锁定DSFID |

1. **I\_get\_status命令**

I\_get\_status 命令码是0x2C，当13.56MHz模块通过串口接收到I\_get\_status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I\_get\_status命令帧结构如表2.35所示。

**表2.35 I\_get\_status命令请求帧格式**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **VData** | | | **FCS** | **EOF** |
| 02 | 10 | XX | 2C | UID | Addr\_Start | Addr\_End | fcs | 03 |

在表2.35请求帧中，VData分为三个部分，第一部分为UID，表明是需要获取哪张卡的相关信息；第二部分为获取块的起始地址；第三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2.36所示。

**表2.36 I\_get\_status命令响应帧格式**

|  |  |  |  |
| --- | --- | --- | --- |
| **CMD** | **Flag** | **VData** | **CRC16/CCITT** |
| 2C | XX | XX … | XX XX |

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，VData的Len = Addr\_End – Addr\_Start + 1，获取安全状态命令示例如表2.37所示。

**表2.37 I\_get\_status命令示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 10 02 2C 76 9D 97 29 00 01 04 E0 00 08 FE 03 | 请求获取00到08地址处的安全状态 |
| Recv | 2C 00 01 00 00 01 01 00 00 01 00 32 38 | 获取成功 |

1. **I\_buzzer命令**

I\_buzzer命令码是0x2D，该命令是对模块上的蜂鸣器操作，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原理机模块通过串口接收到I\_buzzer命令帧结构如表2.38所示。

**表2.38 I\_buzzer命令请求帧格式**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SOF** | **Lenth** | **Flag** | **CMD** | **FCS** | **EOF** |
| 02 | 06 | XX | 2D | 35 | 03 |

13.56MHz原理机模块接收到请求命令后鸣响蜂鸣器，无反馈信息给请求端（如PC），那您可以通过听蜂鸣器是否鸣响来判断是否执行成功，I\_buzzer命令示例如表2.39所示。

**表2.39 I\_buzzer命令示例**

|  |  |  |
| --- | --- | --- |
| **操作** | **请求帧（hex）** | **说明** |
| Send | 02 06 04 2D 37 03 | 请求鸣响蜂鸣器（响一下后自动停止） |

**附录**

FCS（Frame check sequence）：

在原理机15693协议下，每个请求命令帧倒数第二个字节作为该帧的校验和，从SOF（不包含）开始到FCS（不包含）结束的所有字节的和取低八位。计算方法（C语言实现）：

//模256的计算

uint8 calc\_fcs(uint8 \*data)

{

uint8 x,len;

uint8 Result;

len = data[0] - 3 ;

Result = 0;

for ( x = 0; x < len; x++, data++ )

Result = Result + \*data;

return ( Result );

}

说明：参数是要计算的字节数组的起始地址。

CRC-16/CCITT：

原理机模块支持CRC校验，采用CRC-16/CCITT，多项式x16+x12+x5+1。计算方法（C语言实现）：

//CRC16/CCITT的计算

#define ccitt16 0x8408

uint16 CRC16\_CCITT\_Calculate(uint8 \*crc\_data,uint8 crc\_len)

{

uint8 i;

uint16 crc\_value = 0xffff;

while(crc\_len --)

{

crc\_value ^= \*crc\_data ++;

for(i=0;i<8;i++)

{

if(crc\_value & 0x0001) /\*最低位为1，减去除数 \*/

{

crc\_value >>= 1;

crc\_value ^= ccitt16;

}

else /\*最低位为0，不需要减去除数 \*/

{

crc\_value >>= 1; /\*直接移位\*/

}

}

}

return ~crc\_value;

}

说明：第一个参数是要计算的字节数组的起始地址，第二个参数是长度。