Advanced Card Systems Ltd.
Card & Reader Technologies

ACR122S
NFC 读写器(串口)

通信协议 V2.02

info@acs.com.hk
www.acs.com.hk

目录

图目录

表目录

1.0. 简介

ACR122S 是一款非接触式智能卡读写器。它采用串行接口，可读写 ISO 14443-4 A 类和 B 类卡、MIFARE®卡、ISO 18092 或 NFC 卡、以及 FeliCa 标签。本文将介绍用 ACR122S 执行智能卡应用时用到的命令集。

2.0. 特性

- RS-232 串行接口：波特率 = 115200 bps，8-N-1
- USB 接口取电
- 仿 CCID 架构（二进制格式）
- 智能卡读写器：
  - 读/写速率高达 424 kbps
  - 内置天线用于读写非接触式标签，读取智能卡的距离可达 50 mm（视标签的类型而定）
  - 支持 ISO 14443 第 4 部分 A 类和 B 类卡、MIFARE 卡、FeliCa 卡和全部四种 NFC（ISO/IEC 18092）标签
  - 内建防冲突特性（任何时候都只能访问 1 张标签）
  - 符合 ISO 7816 标准的 SAM 卡槽
- 内置外围设备：
  - 2 个用户可控的 LED 指示灯
  - 1 个用户可控的蜂鸣器
- 符合下列标准：
  - ISO 14443
  - ISO18092
  - CE
  - FCC
  - RoHS 2
  - VCCI
  - KC

2.1. 串行接口

ACR122S 通过 RS-232C 串行接口连接主机，波特率为 9600 bps，8-N-1。

引脚	信号	功能
1	**V_{CC}**	为读写器提供 +5 V 的电源（最大 200 mA，常规 100 mA）
2	**TXD**	读写器向主机发送的信号
3	**RXD**	主机向读写器发送的信号
4	**GND**	供电的参考电压等级

表1：引脚配置

2.2. 双色 LED

提供了用户可控的红绿双色 LED 指示灯。

- 如果“卡片接口”尚未建立连接，绿色 LED 闪烁。
- 如果“卡片接口”已建立连接，绿色 LED 亮。
- 如果“卡片接口”正在进行操作，绿色 LED 快速闪烁。
- 红色 LED 仅由应用进行控制。

2.3. 蜂鸣器

提供了用户可控的蜂鸣器，默认处于 OFF 状态。

2.4. SAM 接口

提供了 SAM 卡槽。

2.5. 内置天线

提供了一个中间抽头的 3 圈对称环形天线。

- 预计尺寸 = 60 mm × 48 mm
- 回路电感大概在 1.6 μH 到 2.5 μH 之间
- 标签的工作距离可达到 50 mm（视标签类型而定）。
- 每次只能访问一张标签

3.0. 主机和非接触接口、SAM 及外设之间的通讯

主机通过私有 APDU 访问非接触接口和外设。

另外，主机通过标准 APDU 访问 SAM 接口。

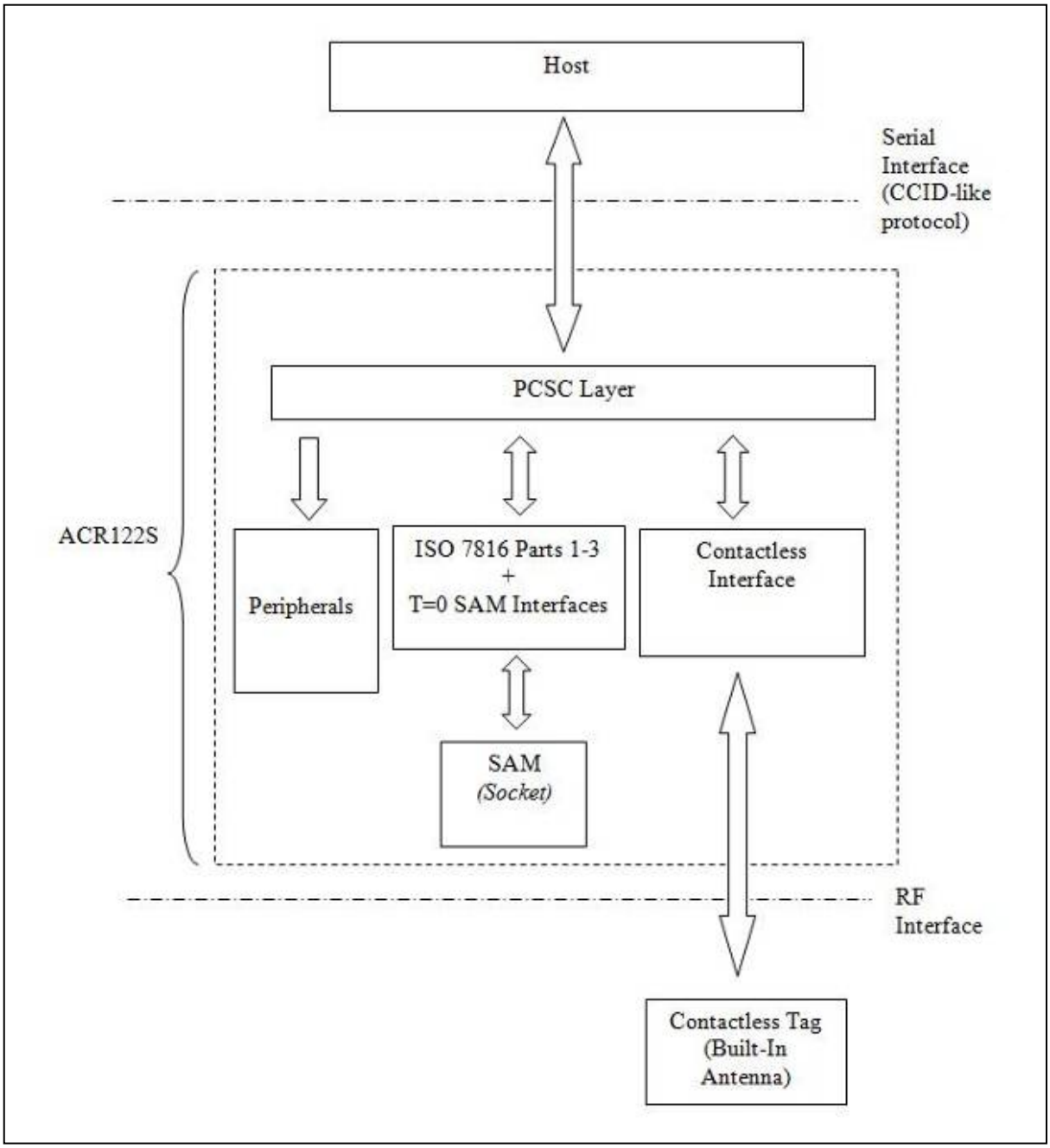


图1： ACR122S 通讯流程图

4.0. 串行接口（仿 CCID 架构）

注：通讯设置为 9600 bps，8-N-1。

主机与 ACR122S 之间的通信协议非常类似于 CCID 协议。

命令帧结构

STX（02h）	Bulk-OUT 头	APDU 命令或参数	校验和	ETX（03h）
1 个字节	10 个字节	M 个字节 （如有）	1 个字节	1 个字节

状态帧结构

STX（02h）	状态	校验和	ETX（03h）
1 个字节	1 个字节	1 个字节	1 个字节

响应帧结构

STX（02h）	Bulk-IN 头	APDU 响应或 abData	校验和	ETX（03h）
1 个字节	10 个字节	N 个字节 （如适用）	1 个字节	1 个字节

校验和 = XOR {Bulk-OUT 头，APDU 命令或参数}

校验和 = XOR {Bulk-IN 头，APDU 响应或 abData}

总体而言，会利用三种 Bulk-OUT 头：

- **HOST_to_RDR_IccPowerOn**：激活 SAM 接口。如果有，将会返回 SAM 的 ATR。
- **HOST_to_RDR_IccPowerOff**：取消激活 SAM 接口。
- **HOST_to_RDR_XfrBlock**：主机和 ACR122S 交换 APDU。

要使用非接触接口和外设，必须先激活 SAM 接口。总之，交换 APDU 必须通过 SAM 接口。

相应的，会用到两种 Bulk-IN 头：

- **RDR_to_HOST_DataBlock**：回应 HOST_to_RDR_IccPowerOn 帧和 HOST_to_RDR_XfrBlock 帧。
- **RDR_to_HOST_SlotStatus**：回应 HOST_to_RDR_IccPowerOff 帧。

RDR = ACR122S; HOST = 主机控制器

HOST_to_RDR = 主机控制器 -> ACR122S

RDR_to_HOST = ACR122S -> 主机控制器

4.1. 协议流举例

A. 激活一个 SAM。

主机			读写器
1. 主机发送一个帧	→	02 62 00 00 00 00 00 01 01 00 00 [校验和] 03	
2. 读写器立即返回一个肯定确认帧。		02 00 00 03（肯定确认帧）	←
		.. 经过一些处理延迟 ..	
3. 读写器返回对命令的响应。		02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [校验和] 03	←

B. 激活一个 SAM（不正确的校验和，主机）。

主机			读写器
1. 主机发送一个已经损坏的帧	→	02 62 00 00 00 00 00 01 01 00 00 [不正确的校验和] 03	
2. 读写器立即返回一个否定确认帧		02 FF FF 03（否定确认帧）	←
3. 主机重新发送帧。	→	02 62 00 00 00 00 00 01 01 00 00 [校验和] 03	
4. 读写器立即返回一个肯定确认帧		02 00 00 03（肯定确认帧）	←
		.. 经过一些处理延迟 ..	
5. 读写器返回对命令的响应。		02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [校验和] 03	←

C. 激活一个 SAM（不正确的校验和，读写器）

	主机		读写器
1. 主机发送一个帧	→	02 62 00 00 00 00 00 01 01 00 00 [校验和] 03	
2. 读写器立即返回一个肯定确认帧		02 00 00 03（肯定确认帧）	←
		.. 经过一些处理延迟 ..	
3. 读写器返回对命令的响应（已经损坏）	→	02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [不正确的校验和] 03	←
4. 主机发送一个 NAK 帧来重新获取响应。		02 00 00 00 00 00 00 00 00 00 00 00 03 (NAK）	
5. 读写器返回对命令的响应		02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [校验和] 03	←

注：*读写器正确接收到主机发送的帧后，会立即发送一个肯定确认帧 = {02 00 00 03} 给主机，通知主机该帧已经被成功接收。主机必须等待命令响应，而读写器在命令处理期间不会接受其它的帧。*

出现差错时，读写器会向主机发送一个否定确认帧，表示帧已经损坏或者格式错误。

校验和错误帧 = {02 FF FF 03}。

长度错误帧 = {02 FE FE 03}。dDwLength 的长度大于 0105h 个字节。

ETX 错误帧 = {02 FD FD 03}。最后一个字节不等于 ETX “03h”。

NAK 帧仅由主机使用来获取最后一个响应。

{02 00 00 00 00 00 00 00 00 00 00 00 03} // 11 个零

5.0. SAM 接口

5.1. 激活 SAM 接口

命令帧结构

STX（02h）	Bulk-OUT 头 （HOST_to_RDR_IccPowerOn）	参数	校验和	ETX（03h）
1 个字节	10 个字节	0 个字节	1 个字节	1 个字节

HOST_to_RDR_IccPowerOn 的结构

偏移	字段	大小	值	说明
0	*bMessageType*	1	62h	-
1	*dDwLength* *<LSB ... MSB>*	4	00000000h	消息特定的数据长度。
5	*bSlot*	1	00-FFh	标识命令的卡槽号。默认为 00h。
6	*bSeq*	1	00-FFh	命令的序列号。
7	*bPowerSelect*	1	00h 01h 02h 03h	施加在 ICC 上的电压值： 00h – 自动电压选择 01h – 5 V 02h – 3 V 03h – 1.8 V
8	*abRFU*	2	-	保留为将来使用

响应帧结构

STX（02h）	Bulk-IN 头 （RDR_to_HOST_DataBlock）	abData	校验和	ETX（03h）
1 个字节	10 个字节	N 个字节（ATR）	1 个字节	1 个字节

RDR_to_HOST_DataBlock 的结构

偏移	字段	大小	值	说明
0	*bMessageType*	1	80h	表示正在从 ACR122S 发送一个数据块。
1	*dwLength* *<LSB ... MSB>*	4	N	*abData* 字段的大小（N 个字节）。
5	*bSlot*	1	与 Bulk-OUT 相同	标识命令的卡槽号。
6	*bSeq*	1	与 Bulk-OUT 相同	相应命令的序列号
7	*bStatus*	1	-	-
8	*bError*	1	-	-

偏移	字段	大小	值	说明
9	*bChainParameter*	1	-	-

例如：激活卡槽 0（默认），序列号为 1，5 V 卡。

HOST -> 02 62 00 00 00 00 00 01 01 00 00 [校验和] 03

RDR -> 02 00 00 03

RDR -> 02 80 0D 00 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [校验和] 03

ATR = 3B 2A 00 80 65 24 B0 00 02 00 82; SW1 SW2 = 90 00

5.2. 取消激活 SAM 接口

命令帧结构

STX（02h）	Bulk-OUT 头 （HOST_to_RDR_IccPowerOff）	参数	校验和	ETX（03h）
1 个字节	10 个字节	0 个字节	1 个字节	1 个字节

HOST_to_RDR_IccPowerOff 的结构

偏移	字段	大小	值	说明
0	*bMessageType*	1	63h	-
1	*dDwLength* *<LSB ... MSB>*	4	00000000h	消息特定的数据长度。
5	*bSlot*	1	00-FFh	标识命令的卡槽号。默认为 00h。
6	*bSeq*	1	00-FFh	命令的序列号。
7	*abRFU*	3	-	保留为将来使用

响应帧结构

STX（02h）	Bulk-IN 头 （RDR_to_HOST_SlotStatus）	abData	校验和	ETX（03h）
1 个字节	10 个字节	0 个字节	1 个字节	1 个字节

RDR_to_HOST_DataBlock 的结构

偏移	字段	大小	值	说明
0	*bMessageType*	1	81h	表示正在从 ACR122S 发送一个数据块。
1	*dwLength* *<LSB ... MSB>*	4	0	*abData* 字段的大小（0 个字节）。
5	*bSlot*	1	与 Bulk-OUT 相同	标识命令的卡槽号。
6	*bSeq*	1	与 Bulk-OUT 相同	相应命令的序列号
7	*bStatus*	1	-	-
8	*bError*	1	-	-
9	*bClockStatus*	1	-	-

例如：取消激活卡槽 0（默认），序列号为 2。

HOST -> 02 63 00 00 00 00 00 02 00 00 00 [校验和] 03

RDR -> 02 00 00 03

RDR -> 02 81 00 00 00 00 00 02 00 00 00 [校验和] 03

5.3. 用 SAM 接口交换数据

命令帧结构

STX（02h）	Bulk-OUT 头 （HOST_to_RDR_XfrBlock）	参数	校验和	ETX（03h）
1 个字节	10 个字节	M 个字节	1 个字节	1 个字节

HOST_to_RDR_XfrBlock 的结构

偏移	字段	大小	值	说明
0	*bMessageType*	1	6Fh	-
1	*dDwLength* *<LSB ... MSB>*	4	M	消息特定的数据长度。
5	*bSlot*	1	00-FFh	标识命令的卡槽号。默认为 00h。
6	*bSeq*	1	00-FFh	命令的序列号。
7	*bBWI*	1	00-FFh	用于延长块的超时等待时间
8	*wLevelParameter*	2	0000h	-

响应帧结构

STX（02h）	Bulk-IN 头 （RDR_to_HOST_DataBlock）	abData	校验和	ETX（03h）
1 个字节	10 个字节	N 个字节（ATR）	1 个字节	1 个字节

RDR_to_HOST_DataBlock 的结构

偏移	字段	大小	值	说明
0	*bMessageType*	1	80h	表示正在从 ACR122S 发送一个数据块。
1	*dwLength* *<LSB ... MSB>*	4	N	*abData* 字段的大小（N 个字节）。
5	*bSlot*	1	与 Bulk-OUT 相同	标识命令的卡槽号。
6	*bSeq*	1	与 Bulk-OUT 相同	相应命令的序列号。
7	*bStatus*	1	-	-
8	*bError*	1	-	-
9	*bChainParameter*	1	-	-

例如：发送 APDU“80 84 00 00 08”给卡槽 0（默认），序列号为 3。

HOST -> 02 6F 05 00 00 00 00 03 00 00 00 80 84 00 00 08 [校验和] 03

RDR -> 02 00 00 03

RDR -> 02 80 0A 00 00 00 00 03 00 00 00 E3 51 B0 FC 88 AA 2D 18 90 00 [校验和] 03

响应 = E3 51 B0 FC 88 AA 2D 18; SW1 SW2 = 90 00

6.0. 用以控制非接触接口和外设的私有 APDU

提供了两个基础命令以控制非接触接口和外设<CLA FFh>。

6.1. 直接传输（Direct Transmit）

此命令用于发送私有 APDU（标签命令），并返回响应数据的长度。

Direct Transmit 的命令结构（标签命令的长度 + 5 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域	
Direct Transmit	FFh	00h	00h	00h	待发送的字节数	标签命令	数据

其中：

Lc 待发送的字节数（1 个字节）。

最大值为 255 个字节

命令数据域 标签命令。

要发送给标签的数据。

Direct Transmit 的响应结构（标签响应 + 数据 + 2 个字节）

项	命令	数据			含义
1	D4 40	Tg		[DataOut[]]	标签交换数据
2	D4 4A	MaxTg	BrTy	[InitiatorData[]]	标签轮询

其中：

Tg 一个字节，包含相关目标的逻辑号。此字节也包含 MI 位（More Information，更多信息），即 bit 6。如果 MI 位设为 1，表示主机控制器需发送更多数据，即 DataOUT[]阵列包含的所有数据。MI 位只对 TPE 目标有效。

DataOut 一个原始数据数组，由非接触芯片发送，0-262 个字节。

MaxTg 非接触芯片初始化目标的最大数量。芯片最多处理两个目标，因此该字段的值不得超过 02h。

Brty 初始化时使用的波特率和调制方式。

00h：106 kbps A 类（ISO/IEC14443 A 类），

01h：212 kbps（FeliCa 轮询），

02h：424 kbps（FeliCa 轮询），

03h：106 kbps B 类（ISO/IEC 14443-3B），

04h：106 kbps Innovision Jewel 标签.

InitiatorData[] 目标初始化时使用的一个数据数组。字段内容因波特率而异。

106 Kbps A 类

字段可选，只有主机控制器需初始化一个 UID 未知的对象时才会出现。

这时，InitiatorData[]含有卡的完整（或部分）UID。如果级联级别为 2 级或 3 级，UID 必须包括级联标签 CT。

级联级别 1

UID1	UID2	UID3	UID4

级联级别 2

UID1	UID2	UID3	UID4	UID5	UID6	UID7

级联级别 3

UID1	UID2	UID3	UID4	UID5	UID6	UID7	UID8	UID9	UID10

106 Kbps B 类

InitiatorData[]结构如下：

AFI（1 个字节）	[轮询方法]

AFI AFI（Application Family Identifier，应用族识别符）表示设备 IC 的目标应用的类型，用在 ATQB 之前预选 PICC。

此字段必选。

轮询方法 此字段可选。表示 ISO/IEC 14443-3B 初始化中用到的方法：

若 bit 0 = 1：ISO/IEC 14443-3B 初始化中的随机方法（选项 1），

若 bit 0 = 0：若 bit 0 = 0：ISO/IEC 14443-3B 初始化中的时间槽方法（选项 2），

如果没有该字段，将使用时间槽法。

212/424 Kbps 此字段必选，包含轮询请求命令（5 个字节，长度字节除外）须用到的完整有效载荷信息。

106 Kbps InnoVision Jewel 标签。本字段无用途。

Data Out 读写器返回的标签响应。

Direct Transmit 的响应结构

响应	响应数据域				
结果	D5 41	状态	[DataIn[]]		SW1 SW2
	D5 4B	NbTg	[TargetData1[]]	[TargetData2[]]	

其中：

状态 一个字节，表示进程是否已成功终止。DEP 或 ISO/IEC 14443-4 PCD 模式下，该字节还表示 *NAD（节点地址）*是否在用以及用 MI 位传输的数据是否完整。

DataIn 一个原始数据数组，由非接触芯片接收，0-262 个字节。

NbTg 初始化目标的数量（最少 0 个，最多 2 个）。

TargetDatai[] TargetDatai[]所含字母“i”是指 “1”或“2”。根据选定的波特率，该字段提供关于检测到的目标信息。以下仅列出一个目标的信息，其他初始化目标的信息与此相同（NbTg 次）。

106 Kbps A 类

Tg	SENS_RES10（2 个字节）	SEL_RES（1 个字节）	NFCIDLength（1 个字节）	NFCID1[]（NFCIDLength bytes）	[ATS[]]（ATSLength bytes11）

106 Kbps B 类

Tg	ATQB 响应（12 个字节）	ATTRIB_RES 的长度（1 个字节）	ATTTRIB_RES[]（ATTRIB_RES 的长度）

212/424 Kbps

Tg	POL_RES 的长度	01h（响应码）	NFCID2t	Pad	SYST_CODE（可选）
1 个字节	1 个字节	1 个字节	8 个字节	8 个字节	2 个字节
	POL_RES（18 或 20 个字节）				

106 Kbps Innovision Jewel 标签

Tg	SENS_RES（2 个字节）	JEWELID[]（4 个字节）

响应数据域　　SW1 SW2。

读卡器返回的状态码。

结果	SW1	SW2	含义
成功	90	00h	操作成功完成。
错误	63	00h	操作失败。
超时错误	63	01h	标签未响应。
校验和错误	63	27h	响应的校验和错误。
参数错误	63	7Fh	标签命令错误。

6.2. 更改通讯速度（Change Communication Speed）

此命令用于更改波特率。

Baud Rate Control 的命令结构（9 个字节）

命令	CLA	INS	P1	P2	Lc
Baud Rate Control	FFh	00h	44h	新波特率	00h

其中：

P2 新波特率

00h = 新波特率设为 9600 bps。

00h = 新波特率设为 115200 bps。

响应数据域 SW1 SW2。

状态码

结果	SW1	SW2	含义
成功	90	当前的波特率	操作成功完成。
错误	63	00h	操作失败。

其中：

SW2 当前的波特率。

00h = 当前的波特率是 9600 bps。

01h = 当前的波特率是 115200 bps。

注：*成功更改通信速度后，程序必须对通信速度进行调整，以便继续进行剩余的数据交换。*

初始通讯速度取决于 R12（0 ohm）是否存在。

- 带有 R12：初始通讯速度 = 115200 bps
- 不带 R12：初始通讯速度 = 9600 bps（默认）

例 1：初始化一个 FeliCa 标签（标签轮询）。

步骤 1. 发送一个“Direct Transmit”APDU。

APDU 命令为“FF 00 00 00 09 D4 4A 01 01 00 FF FF 01 00”

其中，

“Direct Transmit” APDU = “FF 00 00 00”

标签命令的长度 = “09”

标签命令（InListPassiveTarget 212 Kbps） = “D4 4A 01 01”

标签命令（System Code Request）= “00 FF FF 01 00”

发送 APDU 到卡槽 0（默认），序列号为 1。

HOST -> 02 6F 0E 00 00 00 00 01 00 00 00

FF 00 00 00 09 D4 4A 01 01 00 FF FF 01 00

[校验和] 03

RDR -> 02 00 00 03

RDR -> 02 81 1A 00 00 00 00 01 00 00 00

D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00

4B 02 4F 49 8A 8A 80 08 90 00

[校验和] 03

APDU 响应为

“D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00 4B 02 4F 49 8A 8A 80 08 90 00”

其中，

非接触芯片的响应 = “D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00 4B 02 4F 49 8A 8A 80 08”

FeliCa 标签的 NFCID2t = “01 01 05 01 86 04 02 02”

读写器返回的状态码= “90 00”

例 2：向 FeliCa 标签写入 16 个字节（写标签）。

步骤 1. 发送一个“Direct Transmit”APDU。

APDU 命令是“FF 00 00 00 23 D4 40 01 20 08 01 01 05 01 86 04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA”。

其中，

“Direct Transmit” APDU = “FF 00 00 00”

标签命令的长度 = “23”

标签命令（InDataExchange） = “D4 40 01”

标签命令（写数据）= “20 08 01 01 05 01 86 04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA”

发送 APDU 到卡槽 0（默认），序列号为 2。

HOST -> 02 6F 26 00 00 00 00 02 00 00 00

FF 00 00 00 21 D4 40 01 20 08 01 01 05 01 86

04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55

AA 55 AA 55 AA 55 AA

[校验和] 03

RDR -> 02 00 00 03

RDR -> 02 81 11 00 00 00 00 02 00 00 00
D5 41 00 0C 09 01 01 05 01 86 04 02 02 00 00 90 00
[校验和] 03

APDU 响应为“D5 41 00 0C 09 01 01 05 01 86 04 02 02 00 00 90 00”。

其中，

非接触芯片返回的响应 = “D5 41”

FeliCa 标签返回的响应 = “00 0C 09 01 01 05 01 86 04 02 02 00 00”

读写器返回的状态码= “90 00”

例 3：从 FeliCa 标签读取 16 个字节的数据（读标签）。

步骤 1. 发送一个“Direct Transmit”APDU。

APDU 命令为“FF 00 00 00 13 D4 40 01 10 06 01 01 05 01 86 04 02 02 01 09 01 01 80 00”。

其中，

“Direct Transmit” APDU = “FF 00 00 00”

标签命令的长度 = “13”

标签命令（InDataExchange） = “D4 40 01”

标签命令（读数据） = “10 06 01 01 05 01 86 04 02 02 01 09 01 01 80 00”

发送 APDU 到卡槽 0（默认），序列号为 3。

HOST -> 02 6F 18 00 00 00 00 03 00 00 00
FF 00 00 00 13 D4 40 01 10 06 01 01 05 01 86 04
02 02 01 09 01 01 80 00
[校验和] 03
RDR -> 02 00 00 03
RDR -> 02 81 22 00 00 00 00 03 00 00 00
D5 41 00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 90 00
[校验和] 03

APDU 响应：

“D5 41 00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 90 00”

其中，

非接触芯片返回的响应 = “D5 41”

FeliCa 标签返回的响应 =

“00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA”

读写器返回的状态码= “90 00”

例 4：初始化一个 ISO 14443-4 B 类标签（标签轮询）。

步骤 1. 发送一个“Direct Transmit”APDU。

APDU 命令为“FF 00 00 00 05 D4 4A 01 03 00”。

其中，

“Direct Transmit” APDU = “FF 00 00 00”

标签命令的长度 = “05”

标签命令（InListPassiveTarget B 类 106 Kbps） = “D4 4A 01 03 00”

发送 APDU 到卡槽 0（默认），序列号为 4。

HOST -> 02 6F 0A 00 00 00 00 04 00 00 00

FF 00 00 00 05 D4 4A 01 03 00

[校验和] 03

RDR -> 02 00 00 03

RDR -> 02 81 14 00 00 00 00 04 00 00 00

D5 41 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21

90 00 [校验和] 03

APDU 响应为

“D5 4B 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21 90 00”

其中，

非接触芯片返回的响应 = “D5 4B 01 01”

B 类标签的 ATQB = “50 00 01 32 F4 00 00 00 00 33 81 81”

CRC-B = “01 21”

读写器返回的状态码= “90 00”

例 5：发送 APDU 到 ISO 14443-4 B 类标签（数据交换）。

步骤 1. 发送一个“Direct Transmit”APDU。

USER APDU 命令为“00 84 00 00 08”。

APDU 命令为“FF 00 00 00 08 D4 40 01 00 84 00 00 08”

其中，

Direct Transmit APDU = “FF 00 00 00”

标签命令的长度 = “08”

标签命令（InDataExchange） = “D4 40 01”

标签命令（Get Challenge） = “00 84 00 00 08”

发送 APDU 到卡槽 0（默认），序列号为 5。

HOST -> 02 6F 0D 00 00 00 00 05 00 00 00

FF 00 00 00 08 D4 40 01 00 84 00 00 08

[校验和] 03

RDR -> 02 00 00 03

RDR -> 02 81 0F 00 00 00 00 05 00 00 00

D5 41 00 01 02 03 04 05 06 07 08 90 00 90 00

[校验和] 03

APDU 响应为“D5 41 00 0B 01 02 03 04 05 06 07 08 90 00”

其中，

非接触芯片返回的响应 = “D5 41 00”

B 类标签的响应 = “01 02 03 04 05 06 07 08 90 00”

读写器返回的状态码= “90 00”

6.3. 获取固件版本号（Get Firmware Version）

此命令用于获取读写器的固件版本号。

Get Firmware Version 的命令结构（5 个字节）

命令	CLA	INS	P1	P2	Le
Get Response	FFh	00h	48h	00h	00h

其中：

Le　　待获取的字节数（1 个字节）。

最大值为 255 个字节。

Get Firmware Version 的响应结构（10 个字节）

响应	响应数据域
结果	固件版本号

例如：

响应 = 41 43 52 31 32 32 53 31 30 30 = ACR122S100（ASCII）

6.4. 控制双色 LED 和蜂鸣器（Bi-color LED and Buzzer Control）

此命令用于控制双色 LED 指示灯和蜂鸣器的状态。

Bi-color LED and Buzzer Control 的命令结构（9 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域（4 个字节）
Bi-color LED and Buzzer Control	FFh	00h	40h	LED 状态控制	04h	闪烁周期控制

P2 LED 状态控制。

双色 LED 和蜂鸣器的控制结构（1 个字节）

CMD	项	说明
Bit 0	红色 LED 最终状态	1 = 开；0 = 关
Bit 1	绿色 LED 最终状态	1 = 开；0 = 关
Bit 2	红色 LED 状态掩码	1 = 更新其状态 0 = 不更改
Bit 3	绿色 LED 状态掩码	1 = 更新其状态 0 = 不更改
Bit 4	红色 LED 初始闪烁状态	1 = 开；0 = 关
Bit 5	绿色 LED 初始闪烁状态	1 = 开；0 = 关
Bit 6	红色 LED 闪烁掩码	1 = 闪烁 0 = 不闪烁
Bit 7	绿色 LED 闪烁掩码	1 = 闪烁 0 = 不闪烁

命令数据域 闪烁周期控制。

双色 LED 的闪烁周期控制（4 个字节）

字节 0	字节 1	字节 2	字节 3
T1 周期 初始闪烁状态 （单位 = 100 ms）	T2 周期 切换闪烁状态 （单位 = 100 ms）	重复次数	蜂鸣器响应

其中：

字节 3 蜂鸣器响应。在 LED 闪烁期间控制蜂鸣器的状态。

00h = 蜂鸣器不开启。

01h = 蜂鸣器在 T1 周期内开启。

02h = 蜂鸣器在 T2 周期内开启。

03h = 蜂鸣器在 T1 和 T2 周期内开启。

响应数据域　　SW1 SW2。读卡器返回的状态码。

状态码

结果	SW1	SW2	含义
成功	90	LED 当前状态	操作成功完成。
错误	63	00h	操作失败。

LED 当前状态（1 个字节）

状态	项	说明
Bit 0	当前的红色 LED	1 = 开；0 = 关
Bit 1	当前的绿色 LED	1 = 开；0 = 关
Bits 2 – 7	保留	-

注：

1. *LED 状态操作是在 LED 闪烁操作之后进行的。*
2. *如果相应的 LED 状态掩码未启用，则 LED 状态不会发生改变。*
3. *如果相应的 LED 闪烁掩码未启用，则 LED 不会闪烁。同时，重复次数的值必须大于 0。*
4. *T1 和 T2 周期参数主要用于控制 LED 闪烁的工作周期和蜂鸣器的鸣响时间。比如说，如果 T1=1，T2=1，则工作周期 = 50%。工作周期 = T1/（T1 + T2）。*
5. *如果只想控制蜂鸣器，则将 P2"LED 状态控制"置为 0 即可。*
6. *要想使蜂鸣器工作，"重复次数"必须大于 0。*
7. *如果只想控制 LED，则将参数"蜂鸣器响应"置为 0 即可。*

例 1：读取当前 LED 的状态。

// 假设红色和绿色 LED 最初都是关闭状态 //

// 无蜂鸣器响应 //

APDU = "FF 00 40 00 04 00 00 00 00"

响应 = "90 00"。红色和绿色的 LED 均为关闭状态。

例 2：开启红色和绿色的 LED。

// 假设红色和绿色 LED 最初都是关闭状态 //

// 无蜂鸣器响应 //

APDU = "FF 00 40 0F 04 00 00 00 00"

响应 = "90 03"。红色和绿色的 LED 均为开启状态。

将红色和绿色的 LED 都关闭，APDU = "FF 00 40 0C 04 00 00 00 00"

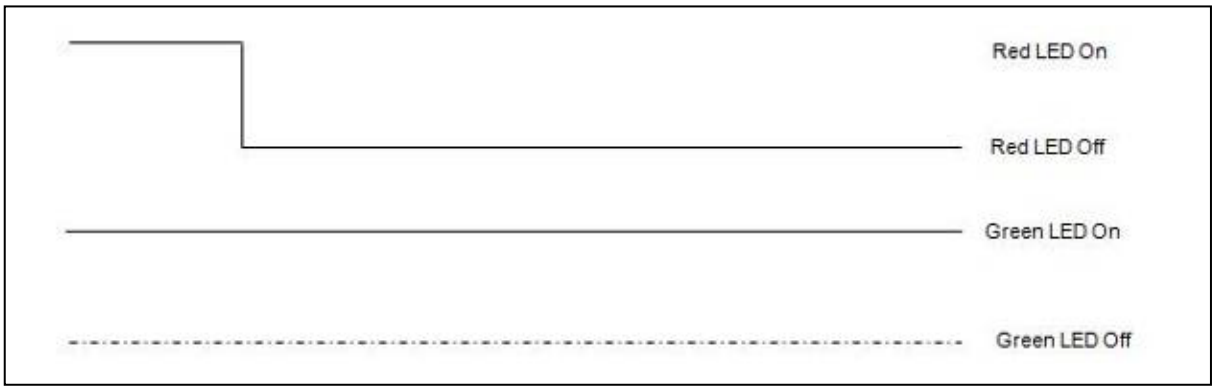
例 3：只关闭红色的 LED，绿色的 LED 保持不变。

// 假设红色和绿色 LED 最初都是开启状态 //

// 无蜂鸣器响应 //

APDU = “FF 00 40 04 04 00 00 00 00”

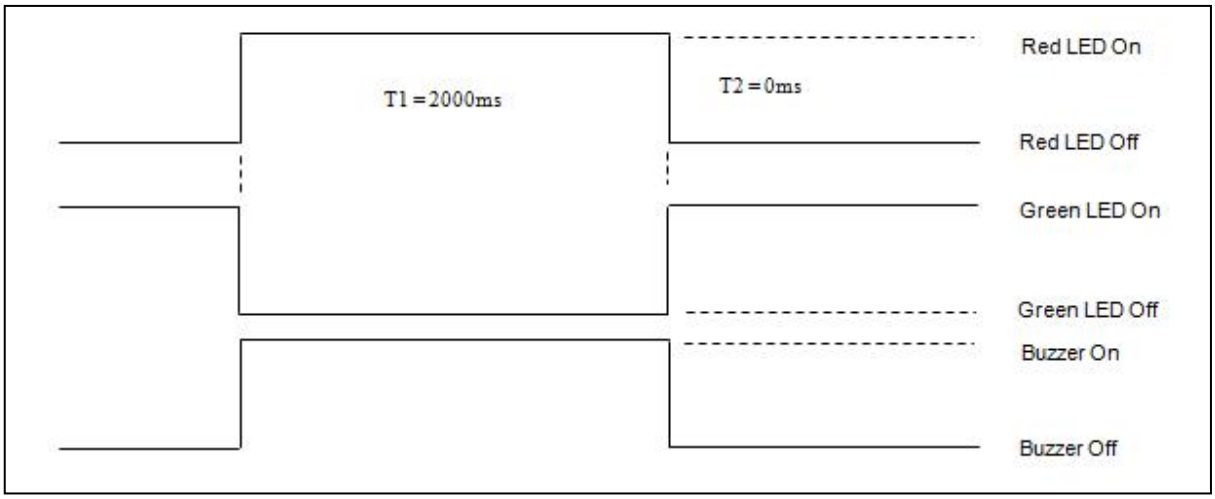
响应 = “90 02”。绿色 LED 保持不变（开启）；红色 LED 关闭，



例 4：将红色的 LED 开启两秒钟。之后返回到初始状态。

// 假设红色 LED 最初是关闭的，而绿色 LED 最初是开启的。//

// 在 T1 周期内，红色 LED 和蜂鸣器会开启，而绿色 LED 会关闭。//



1 Hz = 1000 ms 时间间隔 = 500 ms 开启 + 500 ms 关闭

T1 周期 = 2000 ms = 14h

T2 周期 = 0 ms = 00h

重复次数 = 01h

蜂鸣器响应 = 01h

APDU = “FF 00 40 50 04 14 00 01 01”

响应 = “90 02”

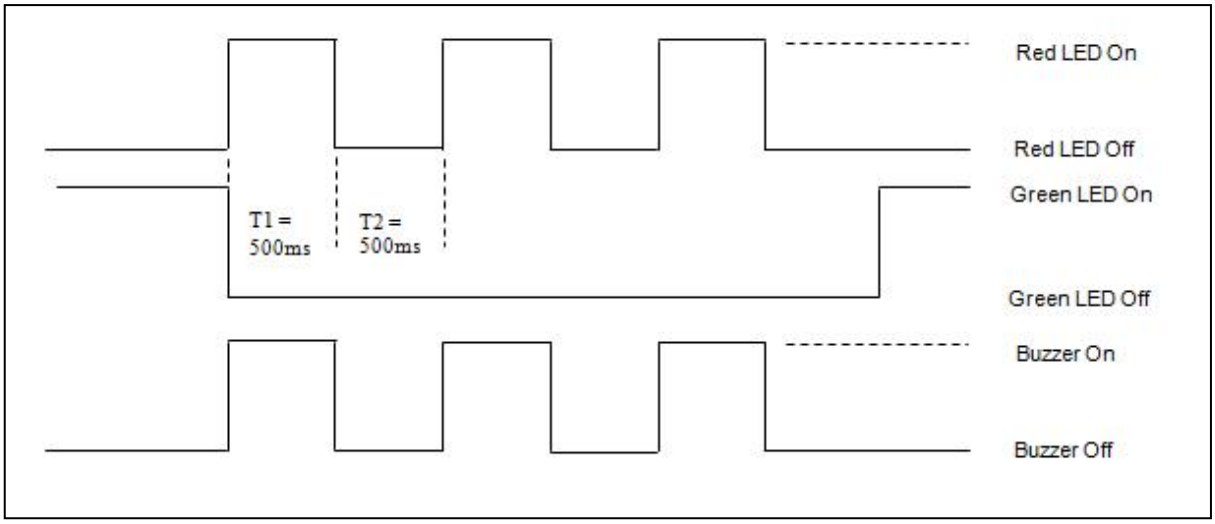
例 5：使红色 LED 闪烁 3 次，每次 1 Hz。之后返回到初始状态。

// 假设红色 LED 最初是关闭的，而绿色 LED 最初是开启的。//

// 红色 LED 最初的闪烁状态是开启的。只有红色 LED 会闪烁。

// 蜂鸣器会在 T1 周期内开启；而绿色 LED 会在 T1 和 T2 周期内关闭。

// 闪烁过后，绿色 LED 会开启。红色 LED 会在闪烁后回到初始状态 //



1 Hz = 1000 ms 时间间隔 = 500 ms 开启 + 500 ms 关闭

T1 周期 = 500 ms = 05h

T2 周期 = 500 ms = 05h

重复次数 = 03h

蜂鸣器响应 = 01h

APDU = “FF 00 40 50 04 05 05 03 01”

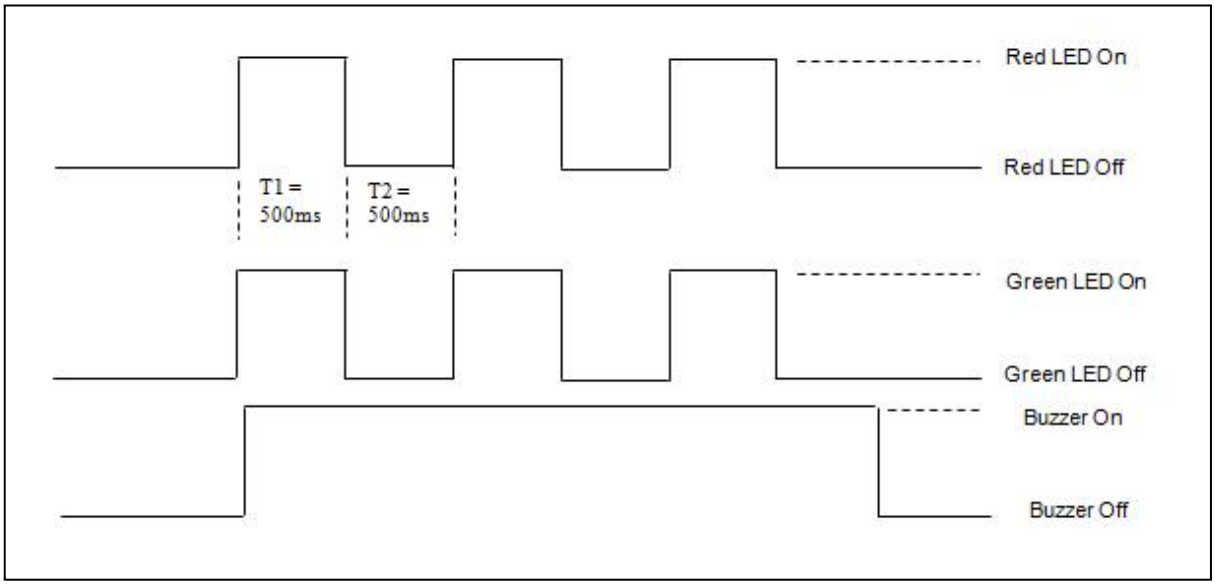
响应 = “90 02”

例 6：使红色和绿色 LED 闪烁 3 次，每次 1 Hz。

// 假设红色 LED 和绿色 LED 最初都是关闭的。//

// 红色 LED 和绿色 LED 的初始闪烁状态都是开启的 //

// 蜂鸣器在 T1 和 T2 周期内都是开启的//



1 Hz = 1000 ms 时间间隔 = 500 ms 开启 + 500 ms 关闭

T1 周期 = 500 ms = 05h

T2 周期 = 500 ms = 05h

重复次数 = 03h

蜂鸣器响应 = 03h

APDU = “FF 00 40 F0 04 05 05 03 03”

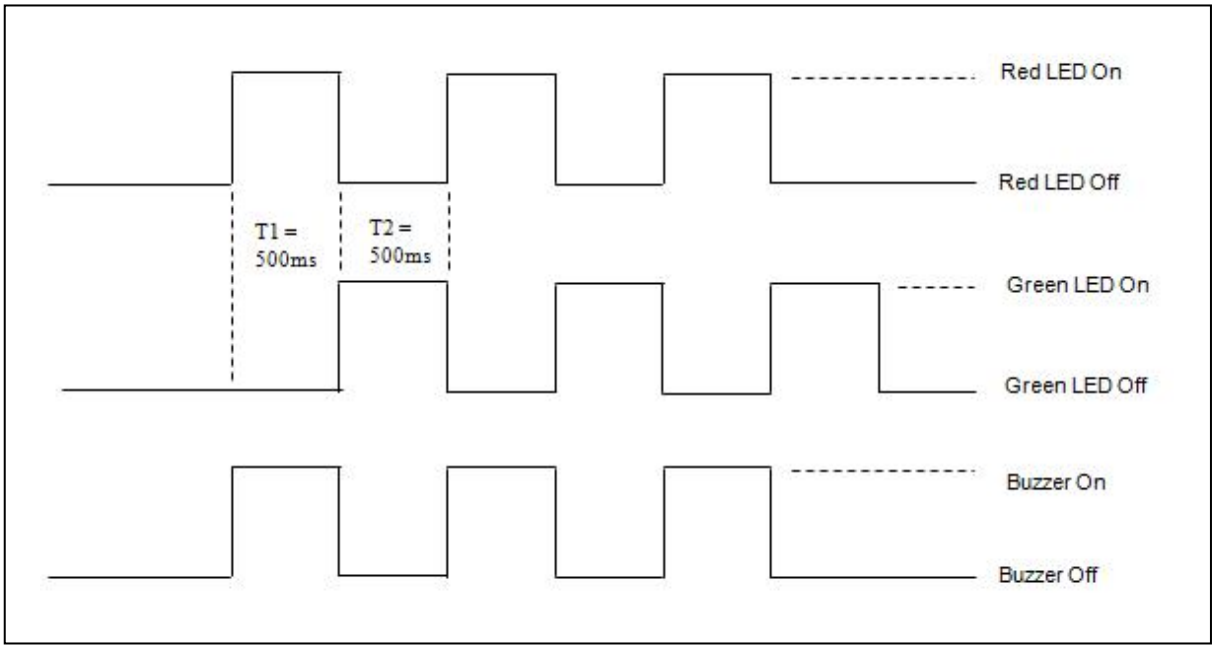
响应 = “90 00”

例 7：使红色和绿色 LED 依次闪烁 3 次，每次 1 Hz。

// 假设红色和绿色 LED 最初都是关闭的。//

// 红色 LED 的初始闪烁状态是开启的；绿色 LED 的初始闪烁状态是关闭的 //

// 蜂鸣器会在 T1 周期内开启//



1 Hz = 1000 ms 时间间隔 = 500 ms 开启 + 500 ms 关闭

T1 周期 = 500 ms = 05h

T2 周期 = 500 ms = 05h

重复次数 = 03h

蜂鸣器响应 = 01h

APDU = “FF 00 40 00 04 05 05 03 01”

响应 = “90 00”

6.5. Topaz512 和 Jewel96

注：*本章节仅适用于固件版本 1.03 的 ACR122S。*

此命令用于进行写-可擦除（8 个字节）、写-不可擦除（8 个字节）、读（8 个字节）以及读数据段的操作。

Topaz 512 and Jewel 96 的命令结构

命令	CLA	INS	P1	P2	Lc	命令数据域
Direct Transmit	FFh	00h	00h	00h	待发送的字节数	标签命令

其中：

Lc 待发送的字节数（1 个字节）。
最大值为 255 个字节。

命令数据域 标签命令。
要发送给标签的数据。

Direct Transmit 的响应结构（响应的长度 + 2 个字节）

响应	响应数据域	
结果	标签响应	SW1 SW2

其中：

Data Out 读写器返回的标签响应。

响应数据域 SW1 SW2。读卡器返回的状态码。

状态码

结果	SW1	SW2	含义
成功	90	00h	操作成功完成。
错误	63	00h	操作失败。

例 1：写-可擦除（8 个字节）Topaz512/Jewel96 标签。

步骤 1. 发送一个“Direct Transmit”APDU。

APDU 命令 “FF 00 00 00 0D D4 40 01 54 05 01 23 45 67 89 AB CD EF”

其中，

“Direct Transmit” APDU = “FF 00 00 00”

标签命令的长度 = “0D”

标签命令（InDataExchange） = “D4 40 01”

标签命令（写-可擦除 8 个字节） = “54”

标签地址（00~3F（16 进制）） = “05“

标签数据 = “01 23 45 67 89 AB CD EF”

发送 APDU 到卡槽 0（默认），序列号为 1。

HOST -> 02 6F 12 00 00 00 00 01 00 00 00
FF 00 00 00 0D D4 40 01 54 05 01 23 45 67 89 AB CD EF
[校验和] 03
RDR -> 02 00 00 03
RDR -> 02 80 0D 00 00 00 00 01 01 00 00
D5 09 05 01 23 45 67 89 AB CD EF 90 00
[校验和] 03

APDU 响应为“D5 09 05 01 23 45 67 89 AB CD EF 90 00”

其中，

非接触芯片返回的响应 = “D5 09 05 01 23 45 67 89 AB CD EF 90 00”

写标签地址 = “05”

写标签 8 个字节数据 = “01 23 45 67 89 AB CD EF”

读写器返回的状态码= “90 00”

例 2：写-不可擦除（8 个字节）Topaz512/Jewel96 标签。

步骤 1. 发送一个“Direct Transmit”APDU。

APDU 命令 “FF 00 00 00 0D D4 40 01 1B 05 FF FF FF FF FF FF FF FF”

其中，

“Direct Transmit” APDU = “FF 00 00 00”

标签命令的长度 = “0D”

标签命令（InDataExchange） = “D4 40 01”

标签命令（写-不可擦除 8 个字节） = “1B”

标签地址（00~3F（16 进制）） = “05“

标签数据 = “FF FF FF FF FF FF FF FF”

发送 APDU 到卡槽 0（默认），序列号为 1。

HOST -> 02 6F 12 00 00 00 00 01 00 00 00
FF 00 00 00 0D D4 40 01 1B 05 FF FF FF FF FF FF FF FF
[校验和] 03
RDR -> 02 00 00 03
RDR -> 02 80 0D 00 00 00 00 01 01 00 00
D5 09 05 FF FF FF FF FF FF FF FF 90 00
[校验和] 03

APDU 响应“D5 09 05 FF FF FF FF FF FF FF FF 90 00”

其中，

非接触芯片返回的响应 = “D5 09 05 FF FF FF FF FF FF FF FF 90 00”

写标签地址 = “05”

写标签 8 个字节数据 = “FF FF FF FF FF FF FF FF”

读写器返回的状态码= “90 00”

例 3：读 8 个字节，一个 Topaz512/Jewel96 标签。

步骤 1. 发送一个“Direct Transmit”APDU。

APDU 命令 “FF 00 00 00 0D D4 40 01 02 05 00 00 00 00 00 00 00 00”

其中，

“Direct Transmit” APDU = “FF 00 00 00”

标签命令的长度 = “0D”

标签命令（InDataExchange） = “D4 40 01”

标签命令（读 8 个字节） = “02”

标签地址（00~3F（16 进制）） = “05“

标签数据 = “00 00 00 00 00 00 00 00”

发送 APDU 到卡槽 0（默认），序列号为 1。

HOST -> 02 6F 12 00 00 00 00 01 00 00 00
FF 00 00 00 0D D4 40 01 02 05 00 00 00 00 00 00 00 00
[校验和] 03
RDR -> 02 00 00 03

RDR -> 02 80 0D 00 00 00 00 01 01 00 00
D5 09 05 01 23 45 67 89 AB CD EF 90 00
[校验和] 03

APDU 响应为“D5 09 05 01 23 45 67 89 AB CD EF 90 00”

其中，

非接触芯片返回的响应 = “D5 09 05 01 23 45 67 89 AB CD EF 90 00”

读标签地址 = “05”

读标签 8 个字节数据 = “01 23 45 67 89 AB CD EF”

读写器返回的状态码= “90 00”

例 4：读数据段，一个 Topaz512/Jewel96 标签。

步骤 1. 发送一个“Direct Transmit”APDU。

APDU 命令 “FF 00 00 00 0D D4 40 01 10 00 00 00 00 00 00 00 00”

其中，

“Direct Transmit” APDU = “FF 00 00 00”

标签命令的长度 = “0D”

标签命令（InDataExchange） = “D4 40 01”

标签命令（读数据段） = “10”

标签地址（00/10/20/30）= “00 “（Block 0）

标签数据 = “00 00 00 00 00 00 00 00”

发送 APDU 到卡槽 0（默认），序列号为 1。

HOST -> 02 6F 12 00 00 00 00 01 00 00 00
FF 00 00 00 0D D4 40 01 10 00 00 00 00 00 00 00 00
[校验和] 03
RDR -> 02 00 00 03
RDR -> 02 80 0D 00 00 00 00 01 01 00 00
D5 41 00 ...<128 个字节数据> ...90 00
[校验和] 03

APDU 响应“D5 41 00 … <128 个字节数据> … 90 00”

其中，

非接触芯片返回的响应 = “D5 41 00 … <128 bytes data> … 90 00”

读标签数据段数据 = “<128 个字节数据>”

读写器返回的状态码= “90 00”

例 5：在 Topaz/Jewel 写入多数据。

注：*此功能只限在数据段 0 写数据。*

步骤 1. 发送一个“Direct Transmit”APDU。

APDU 命令 “FF 00 00 00 36 D4 40 01 58 20 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47”

其中，

“Direct Transmit” APDU = “FF 00 00 00”

标签命令的长度 = “36”

标签命令（InDataExchange） = “D4 40 01”

标签命令（写多数据） = “58”

标签地址 = “20（0 0100 000）“（块 4, 字节-0）（参见 **图 2**）

标签数据 = “00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47”

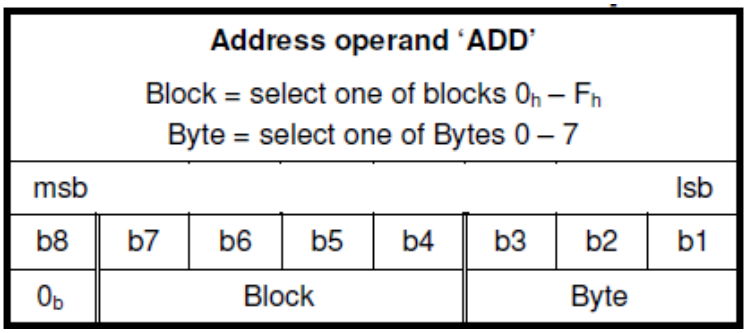


图2：标签地址 “ADD”

发送 APDU 到卡槽 0（默认），序列号为 1。

HOST -> 02 6F 3B 00 00 00 00 01 00 00 00

FF 00 00 00 36 D4 40 01 58 20 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

[校验和] 03

RDR -> 02 00 00 03

RDR -> 02 80 05 00 00 00 00 01 01 00 00

D5 41 00 90 00

[校验和] 03

APDU 响应 “D5 41 00 90 00”

其中，

非接触芯片返回的响应 = “D5 41 00 90 00”

写标签数据 = “00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47”

读写器返回的状态码= “90 00”

如果读写器返回的状态码 = “63 00”，表示操作未完成。

例 6：在 Topaz512/Jewel96 标签写入多个 8 字节的数据。

步骤 1. 发送一个“Direct Transmit”APDU。

APDU 命令 “FF 00 00 00 36 D4 40 01 5A 04 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47”

其中，

“Direct Transmit” APDU = “FF 00 00 00”

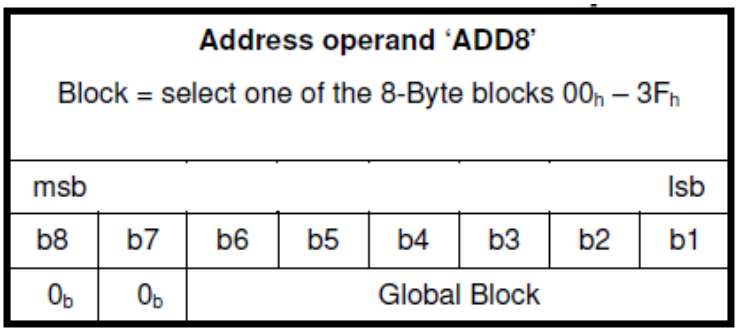
标签命令的长度 = “36”

标签命令（InDataExchange） = “D4 40 01”

标签命令（写多数据） = “5A”

标签地址（块 No.00-3F）= “04 “（块 No. 4）（参见**图 3**）

标签数据 = “00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47”



Address operand ‘ADD8’ Block = select one of the 8-Byte blocks 00_h – $3F_h$							
msb							lsb
b8	b7	b6	b5	b4	b3	b2	b1
0_b	0_b	Global Block					

图3： 标签地址 “ADD8”

发送 APDU 到卡槽 0（默认），序列号为 1。

HOST -> 02 6F 3B 00 00 00 00 01 00 00 00

FF 00 00 00 36 D4 40 01 5A 04 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

[校验和] 03

RDR -> 02 00 00 03

RDR -> 02 80 04 00 00 00 00 01 01 00 00

D5 09 90 00

[校验和] 03

APDU 响应 “D5 09 90 00”

其中，

非接触芯片返回的响应 = “D5 09 90 00”

写标签数据 = “00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47”

读写器返回的状态码= “90 00”

如果读写器返回的状态码 = “63 00”，表示操作未完成。

6.6. ISO 14443-4 A 类和 B 类标签的基本流程

典型的操作顺序为：

1. 用正确的参数（A 类或 B 类）扫描天线场内的标签（轮询）。
2. 更改波特率（此选项仅对 A 类标签可选）。
3. 执行任意 T=CL 命令。
4. 取消选择标签。

步骤 1. 设置重试次数。

HOST -> 02 6F 0B 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 06 D4 32 05 00 00 01 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 04 00 00 00 00 01 01 00 00

RDR -> D5 33 90 00 [校验和] 03

步骤 2. 轮询 ISO 14443-4 A 类标签，106 kbps。

HOST -> 02 6F 09 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 04 D4 4A 01 00 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 15 00 00 00 00 01 01 00 00

RDR -> D5 4B 01 01 00 08 28 04 85 82 2F A0 07 77 F7 80 02 47 65 90 00 [校验和] 03

其中， 查找到的标签数量 = [01];

目标编号 = 01

SENS_RES = 00 08; SEL_RES = 28，

UID 长度 = 4；

UID = 85 82 2F A0

ATS = 07 77 F7 80 02 47 65

操作完成 = 90 00

或

步骤 2. 轮询 ISO 14443-4 B 类标签，106 kbps。

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 05 D4 4A 01 03 00 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 14 00 00 00 00 01 01 00 00

RDR -> D5 4B 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21 90 00 [校验和] 03

其中， 查找到的标签数量 = [01];

目标编号 = 01

ATQB = 50 00 01 32 3B 00 00 00 00 33 81 81。

ATTRIB_RES 长度 = 01; ATTRIB_RES = 21

操作完成 = 90 00

步骤 3. 将波特率由默认值改为其他值（可选）。

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 05 D4 4E 01 02 02 [校验和] 03 // 波特率更改成 424 kbps

或

HOST -> FF 00 00 00 05 D4 4E 01 01 01 [校验和] 03 // 波特率更改成 212 kbps

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 4F [00] 90 00 [校验和] 03

注：*请检查标签支持的最大波特率。仅支持 A 类标签。*

步骤 4. 执行 T=CL 命令，APDU 随机取数 = 00 84 00 00 08。

HOST -> 02 6F 0D 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 08 D4 40 01 00 84 00 00 08 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 0F 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 62 89 99 ED C0 57 69 2B 90 00 90 00 [校验和] 03

其中，响应数据 = 62 89 99 ED C0 57 69 2B 90 00

步骤 5. 取消选择标签。

HOST -> 02 6F 08 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 03 D4 44 01 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

步骤 6. 关闭天线电源（可选）。

HOST -> 02 6F 09 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 04 D4 32 01 00

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 04 00 00 00 00 01 01 00 00

RDR -> D5 33 90 00 [校验和] 03

注：*更多详情请参阅标签标准。*

6.7. MIFARE 应用的基本流程

典型的操作顺序为：

1. 扫描天线场内的标签（轮询）。
2. 认证。
3. 读/写标签的存储内容。
4. 终止标签（可选）。

步骤 1. 设置重试次数。

HOST -> 02 6F 0B 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 06 D4 32 05 00 00 01 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 04 00 00 00 00 01 01 00 00

RDR -> D5 33 90 00 [校验和] 03

步骤 2. 轮询 MIFARE 1K/4K 标签，106 kbps。

HOST -> 02 6F 09 00 00 00 00 01 00 00 00 FF 00 00 00 04 D4 4A 01 00 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 0E 00 00 00 00 01 01 00 00

RDR -> D5 4B 01 01 00 02 18 04 F6 8E 2A 99 90 00 [校验和] 03

其中， 查找到的标签数量 = [01]; 目标编号 = 01

SENS_RES = 00 02; SEL_RES = 18，

UID 长度 = 4；

UID = F6 8E 2A 99

操作完成 = 90 00

注：*可通过识别 SEL_RES 来鉴定标签类型。*

几种常见标签类型的 SEL_RES

00 = MIFARE Ultralight®

08 = MIFARE Classic® 1K

09 = MIFARE® Mini

18 = MIFARE Classic® 4K

20 = MIFARE® DESFire®

28 = JCOP30

98 = Gemplus MPCOS

步骤 3. 认证密钥 A，Block **04**，KEY = FF FF FF FF FF FF，UID = F6 8E 2A 99。

HOST -> 02 6F 14 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 0F D4 40 01 **60** **04** FF FF FF FF FF FF F6 8E 2A 99 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

***注：**如果认证失败，显示错误码 [XX]。*

[00] = 有效，其他 = 错误。更多详情请参阅错误码表。

认证密钥 B 时

HOST -> 02 6F 14 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 0F D4 40 01 **61** **04** FF FF FF FF FF FF F6 8E 2A 99

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

步骤 4. 读取值块 04 的内容。

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00 FF 00 00 00 05 D4 40 01 **30** **04** [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 15 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [校验和] 03

其中，值块数据 = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

步骤 5. 更新 Block **04** 的内容。

HOST -> 02 6F 1A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 15 D4 40 01 **A0** **04** 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

步骤 6. 终止标签（可选）。

HOST -> 02 6F 08 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 03 D4 44 01 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 45 [00] 90 00 [校验和] 03

6.7.1. 处理 MIFARE Classic 1K/4K 标签的值块

值块有电子钱包的功能，例如增值，减值，恢复，传输等。值块的固定数据结构使得值块可进行差错检验，差错校正和备份管理。

字节号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
说明	值				值				值				地址	地址	地址	地址

其中：

值 表示一个有符号的 4 字节值。其中值的最低有效字节存储在最低地址字节内。取反的字节以标准 2 的补码格式保存。

Adr 表示一个 1 字节地址，可用于保存一个块的存储地址。（可选）

例如：

值 100（十进制） = 64（十六进制），假设 Block = 05h

格式化的值块 = 64 00 00 00 9B FF FF FF 64 00 00 00 05 FA 05 FA

步骤 1. **用值 100（十进制）更新**值块 **05** 的内容。

HOST -> 02 6F 1A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 15 D4 40 01 **A0** **05** 64 00 00 00 9B FF FF FF 64 00 00 00 05 FA 05 FA [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

步骤 2. // 使值块 **05** 的值增加 **1（十进制）**。

HOST -> 02 6F 0E 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 09 D4 40 01 **C1** **05** 01 00 00 00 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

***注：** // 使值块 05 的值减少 1（十进制）。*

HOST -> FF 00 00 00 09 D4 40 01 C0 05 01 00 00 00

步骤 3. 传输值块 **05** 先前计算的值**（十进制）**.

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 05 D4 40 01 **B0** **05** [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

***注：**恢复值块 **05** 的值（取消先前的增值或减值操作）。*

HOST -> FF 00 00 00 05 D4 40 01 **C2 05**

步骤 4. 读取值块 **05** 的内容。

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 05 D4 40 01 **30 05** [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 15 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 65 00 00 00 9A FF FF FF 65 00 00 00 05 FA 05 FA 90 00 [校验和] 03

其中，值 = 101（十进制）

步骤 5. 复制值块 **05** 的值到值块 **06（十进制）**。

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 05 D4 40 01 **C2 05** [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

HOST ->02 6F 0A 00 00 00 00 01 00 00 00

HOST ->FF 00 00 00 05 D4 40 01 **B0 06** [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

步骤 6. 读取值块 **06** 的内容。

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 05 D4 40 01 **30 06** [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 15 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 65 00 00 00 9A FF FF FF 65 00 00 00 05 FA 05 FA 90 00 [校验和] 03

其中，值 = 101（十进制）地址“05 FA 05 FA”表明该值是从值块 05 复制的。

注：*更多详情请参阅 MIFARE 标准。*

扇区 （共 16 个扇区，每个扇区包含 4 个连续的块）	数据块 （3 个块，每块 16 个字节）	尾部块 （1 个块，16 个字节）
扇区 0	00h ~ 02h	03h
扇区 1	04h ~ 06h	07h
..		
..		
扇区 14	38h ~ 0Ah	3Bh
扇区 15	3Ch ~ 3Eh	3Fh

1 KB

表2： MIFARE 1K 卡的内存结构

扇区 （共 32 个扇区，每个扇区包含 4 个连续的块）	数据块 （3 个块，每块 16 个字节）	尾部块 （1 个块，16 个字节）
扇区 0	00h ~ 02h	03h
扇区 1	04h ~ 06h	07h
..		
..		
扇区 30	78h ~ 7Ah	7Bh
扇区 31	7Ch ~ 7Eh	7Fh

2 KB

扇区 （共 8 个扇区，每个扇区包含 16 个连续的块）	数据块 （15 个块，每块 16 个字节）	尾部块 （1 个块，16 个字节）
扇区 32	80h ~ 8Eh	8Fh
扇区 33	90h ~ 9Eh	9Fh
..		
..		
扇区 38	E0h ~ EEh	EFh
扇区 39	F0h ~ FEh	FFh

2K 字节

表3： MIFARE 4K 卡的内存结构

注：*一旦认证成功，可自由访问同一扇区的所有数据块。例如，扇区 1 的数据块 04h 认证成功后，就可自由访问扇区 1 的其他数据块 04h ~ 07h。*

6.7.2. 访问 MIFARE Ultralight 标签

典型的操作顺序为：

1. 扫描天线场内的标签（轮询）。
2. 读/写标签的存储内容。
3. 终止标签（可选）。

步骤 1. 设置重试次数。

HOST -> 02 6F 0B 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 06 D4 32 05 00 00 01 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 04 00 00 00 00 01 01 00 00

RDR -> D5 33 90 00 [校验和] 03

步骤 2. 轮询 MIFARE Ultralight 标签，106 kbps。

HOST -> 02 6F 09 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 04 D4 4A 01 00 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 11 00 00 00 00 01 01 00 00

RDR -> D5 4B 01 01 00 44 00 07 04 6E 0C A1 BF 02 84 90 00 [校验和] 03

其中， 查找到的标签数量 = [01];

目标编号 = 01

SENS_RES = 00 44; SEL_RES = 00，

UID 长度 = 7；

UID = 04 6E 0C A1 BF 02 84

操作完成 = 90 00

步骤 3. 读取页 **04** 的内容。

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 05 D4 40 01 **30 04** [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 15 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [校验和] 03

其中，值块数据 = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

注：*将获取 4 个连续页。//将获取页 4、5、6 和 7。每个数据页包括 4 个字节。*

步骤 4. 用数据“AA BB CC DD”更新页 **04** 的内容。

HOST -> 02 6F 0E 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 09 D4 40 01 **A2** **04** AA BB CC DD [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

OR

步骤 4. 用数据“AA BB CC DD”写（MIFARE 兼容的写功能）页 04 的内容。

HOST -> 02 6F 1A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 15 D4 40 01 **A0** **04** AA BB CC DD 00 00 00 00 00 00 00 00 00 00 00 00 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

注：*此命令用于适应已建立的 MIFARE Classic 1K/4K 卡的基础结构。数据须组装成一个 16 字节的帧。前 4 个字节是数据，其余字节（12 个 0）是填充。虽然发送给读写器的是 16 个字节，但是只更新页 4（4 个字节）。*

步骤 5. 再次读取页 **04** 的内容。

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 05 D4 40 01 **30** **04** [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 15 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [校验和] 03

其中，值块数据 = AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16

注：*只更新页 4。页 5、6、7 保持不变。*

步骤 6. 终止标签（可选）。

HOST -> 02 6F 08 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 03 D4 44 01 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 45 [00] 90 00 [校验和] 03

注：*更多详情请参阅 MIFARE Ultralight 标准。*

字节号	0	1	2	3	页
序列号	SN0	SN1	SN2	BCC0	0
序列号	SN3	SN4	SN5	SN6	1
内部 / 锁	BCC1	Internal	Lock0	Lock1	2
OTP	OPT0	OPT1	OTP2	OTP3	3
数据读/写	Data0	Data1	Data2	Data3	4
数据读/写	Data4	Data5	Data6	Data7	5
数据读/写	Data8	Data9	Data10	Data11	6
数据读/写	Data12	Data13	Data14	Data15	7
数据读/写	Data16	Data17	Data18	Data19	8
数据读/写	Data20	Data21	Data22	Data23	9
数据读/写	Data24	Data25	Data26	Data27	10
数据读/写	Data28	Data29	Data30	Data31	11
数据读/写	Data32	Data33	Data34	Data35	12
数据读/写	Data36	Data37	Data38	Data39	13
数据读/写	Data40	Data41	Data42	Data43	14
数据读/写	Data44	Data45	Data46	Data47	15

512 位
或
64 个字节

表4： MIFARE Ultralight 卡的内存结构

6.7.3. 访问 MIFARE Ultralight C 标签

典型的操作顺序为：

1. 扫描天线场内的标签（轮询）。
2. 认证。
3. 读/写标签的存储内容。
4. 终止标签（可选）。

步骤 1. 设置重试次数。

HOST -> 02 6F 0B 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 06 D4 32 05 00 00 01 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 04 00 00 00 00 01 01 00 00

RDR -> D5 33 90 00 [校验和] 03

步骤 2. 轮询 MIFARE Ultralight C 标签，106 kbps

HOST -> 02 6F 09 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 04 D4 4A 01 00 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 11 00 00 00 00 01 01 00 00

RDR -> D5 4B 01 01 00 44 00 07 04 6E 0C A1 BF 02 84 90 00 [校验和] 03

其中， 查找到的标签数量 = [01]; 目标编号 = 01

SENS_RES = 00 44; SEL_RES = 00，

UID 长度 = 7；

UID = 04 6E 0C A1 BF 02 84

操作完成 = 90 00

步骤 3. 3DES 认证

HOST -> 02 6F 09 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 04 D4 42 1A 00 10 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 0E 00 00 00 00 01 01 00 00

RDR -> D5 43 [00] 04 77 64 89 99 74 24 67 90 00 [校验和] 03

其中，卡片的 3DES 随机数 = [04 77 64 89 99 74 24 67];

h = 90 00

HOST -> 02 6F 18 00 00 00 00 01 00 00 00

HOST ->FF 00 00 00 13 D4 42 AF 88 68 45 07 65 86 99 67 00 53 77 56 98 65 49 67 [校验和] 03

其中，卡片接收的 3DES 响应 = [88 68 45 07 65 86 99 67 00 53 77 56 98 65 49 67];

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 0E 00 00 00 00 01 01 00 00

RDR -> D5 43 [00] 00 06 78 53 80 68 89 61 24 90 00 [校验和] 03

其中，卡片发送的 3DES 响应 = [06 78 53 80 68 89 61 24];

操作完成 = 90 00

注：*须检查卡片发送的 3DES 响应以保证卡片合法。*

步骤 4. 读取页 **04** 的内容。

HOST -> 02 6F 09 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 05 D4 40 01 **30 04** [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 15 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [校验和] 03

其中，值块数据 = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

注：*将获取 4 个连续页。//获取页 4、5、6 和 7。每个数据页包括 4 个字节。*

步骤 5. 用数据“AA BB CC DD”更新页 **04** 的内容。

HOST -> 02 6F 0E 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 09 D4 40 01 **A2 04** AA BB CC DD [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] 90 00 [校验和] 03

OR

步骤 5. 用数据“AA BB CC DD”写（MIFARE 兼容的写功能）页 **04 的内容**。

HOST -> 02 6F 1A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 15 D4 40 01 **A0 04** AA BB CC DD 00 00 00 00 00 00 00 00 00 00 00 00 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR ->D5 41 [00] 90 00 [校验和] 03

注：*此命令用于适应已建立的 MIFARE Classic 1K/4K 卡的基础结构。数据须组装成一个 16 字节的帧。前 4 个字节是数据，其余字节（12 个 0）是填充。虽然发送给读写器的是 16 个字节，但是只更新页 4（4 个字节）。*

步骤 6.再次读取页 **04** 的内容。

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 05 D4 40 01 **30 04** [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 15 00 00 00 00 01 01 00 00

RDR -> D5 41 [00] AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [校验和] 03

其中，值块数据 = AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16

注：*只更新页 4。页 5、6、7 保持不变。*

步骤 7.终止标签（可选）。

HOST -> 02 6F 08 00 00 00 00 01 00 00 00

HOST -> FF 00 00 00 03 D4 44 01 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 05 00 00 00 00 01 01 00 00

RDR -> D5 45 [00] 90 00 [校验和] 03

注：*更多详情请参阅 MIFARE Ultralight C 标准。*

字节号	0	1	2	3	页
序列号	SN0	SN1	SN2	BCC0	0
序列号	SN3	SN4	SN5	SN6	1
内部 / 锁	BCC1	Internal	锁	锁	2
OTP	OTP0	OTP1	OTP2	OTP3	3
数据读/写	Data0	Data1	Data2	Data3	4
数据读/写	Data4	Data5	Data6	Data7	5
数据读/写	Data8	Data9	Data10	Data11	6
数据读/写	Data12	Data13	Data14	Data15	7
数据读/写	Data16	Data17	Data18	Data19	8
数据读/写	Data20	Data21	Data22	Data23	9
数据读/写	Data24	Data25	Data26	Data27	10
数据读/写	Data28	Data29	Data30	Data31	11
数据读/写	Data32	Data33	Data34	Data35	12

字节号	0	1	2	3	页
数据读/写	Data36	Data37	Data38	Data39	13
数据读/写	Data40	Data41	Data42	Data43	14
数据读/写	Data44	Data45	Data46	Data47	15
数据读/写	Data48	Data49	Data50	Data51	16
数据读/写	Data52	Data53	Data54	Data55	17
数据读/写	Data56	Data57	Data58	Data59	18
数据读/写	Data60	Data61	Data62	Data63	19
数据读/写	Data64	Data65	Data66	Data67	20
数据读/写	Data68	Data69	Data70	Data71	21
数据读/写	Data72	Data73	Data74	Data75	22
数据读/写	Data76	Data77	Data78	Data79	23
数据读/写	Data80	Data81	Data82	Data83	24
数据读/写	Data84	Data85	Data86	Data87	25
数据读/写	Data88	Data89	Data90	Data91	26
数据读/写	Data92	Data93	Data94	Data95	27
数据读/写	Data96	Data97	Data98	Data99	28
数据读/写	Data100	Data101	Data102	Data103	29
数据读/写	Data104	Data105	Data106	Data107	30
数据读/写	Data108	Data109	Data110	Data111	31
数据读/写	Data112	Data113	Data114	Data115	32
数据读/写	Data116	Data117	Data118	Data119	33
数据读/写	Data120	Data121	Data122	Data123	34
数据读/写	Data124	Data125	Data126	Data127	35
数据读/写	Data128	Data129	Data130	Data131	36
数据读/写	Data132	Data133	Data134	Data135	37
数据读/写	Data136	Data137	Data138	Data139	38
数据读/写	Data140	Data141	Data142	Data143	39
锁	锁	锁	-	-	40
16 位计数器	16 位计数器	16 位计数器	-	-	41
认证配置	认证配置	认证配置	认证配置	认证配置	42
认证配置	认证配置	认证配置	认证配置	认证配置	43
认证密钥	认证密钥	认证密钥	认证密钥	认证密钥	44
认证密钥	认证密钥	认证密钥	认证密钥	认证密钥	45
认证密钥	认证密钥	认证密钥	认证密钥	认证密钥	46

字节号	0	1	2	3	页
认证密钥	认证密钥	认证密钥	认证密钥	认证密钥	47

表5：MIFARE Ultralight C 卡的内存结构

页总大小：198 字节的 792 位.

6.8. FeliCa 应用的基本流程

步骤 0. 启动应用程序，首先要激活“SAM 接口”。返回 SAM 的 ATR（如果插入了 SAM）或者一个私有 ATR“3B 00”（加入没有插入 SAM）。换句话说，从应用的角度看，SAM 总是存在。

步骤 1. 其次是修改非接触芯片的操作参数。设置重试次数为 2。

步骤 2. 发送两个 APDU“Direct Transmit”和“Get Response”，以轮询 FeliCa 标签（标签轮询）。

步骤 3. 如果没有发现标签，返回步骤 2，直到发现一个 FeliCa 标签。

步骤 4. 发送一个 APDU（读或写标签）访问 FeliCa 标签。

步骤 5. 如果不对 FeliCa 标签执行任何操作，则返回步骤 2 轮询其他 FeliCa 标签。

..

步骤 N. 取消激活“SAM 接口”。关闭应用程序。

注：

1. *标签命令“InListPassiveTarget”的默认重试次数是无限次。发送 APDU“FF 00 00 00 06 D4 32 05 00 00 01”以修改重试次数为 2。*
2. *如果不用访问非接触标签，建议关闭天线。*

   用以开启天线电源的 APDU = APDU “FF 00 00 00 04 D4 32 01 03”

   用以关闭天线电源的 APDU = APDU “FF 00 00 00 04 D4 32 01 02”

6.9. NFC 论坛 Type 1 标签应用的基本流程

例如：Jewel 和 Topaz 标签。

典型的操作顺序为：

1. 扫描天线场内的标签（轮询）。
2. 读取/更新标签的存储内容。
3. 取消选择标签。

步骤 1. 设置重试次数。

HOST -> 02 6F 0B 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 06 D4 32 05 00 00 01 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 04 00 00 00 00 01 01 00 00

RDR -> D5 33 90 00 [校验和] 03

步骤 2. 轮询 Jewel 或 Topaz 标签，106 kbps

HOST -> 02 6F 09 00 00 00 00 01 00 00 00（HOST_to_RDR_XfrBlock 结构）

HOST -> FF 00 00 00 04 D4 4A 01 04 [校验和] 03

RDR -> 02 00 00 03（等待标签）

RDR -> 02 80 0C 00 00 00 00 01 01 00 00

RDR -> D5 4B 01 01 0C 00 B5 3E 21 00 90 00 [校验和] 03

其中， 查找到的标签数量 = [01];

目标编号 = 01

ATQA_RES = 0C 00; UID = B5 3E 21 00

操作完成 = 90 00

步骤 3. 读存储地址 08（块 1：字节-0）

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00 FF 00 00 00 05 D4 40 01 01 08 [校验和] 03

RDR -> 02 00 00 03 02 80 06 00 00 00 00 01 01 00 00 D5 41 [00] 18 90 00 [校验和] 03

其中，响应数据 = 18

注：*从存储地址 00 开始读取标签的所有存储内容。*

HOST -> 02 6F 09 00 00 00 00 01 00 00 00 FF 00 00 00 04 D4 40 01 00 [校验和] 03

RDR -> 02 00 00 03 02 80 7F 00 00 00 00 01 01 00 00 D5 41 00 11 48

RDR -> show all data … 90 00 [校验和] 03

步骤 4. 更新存储地址 08（Block 1：Byte-0）更新为数据 FF。

HOST -> 2 6F 0B 00 00 00 00 01 00 00 00 FF 00 00 00 06 D4 40 01 53 08 FF [校验和] 03

RDR -> 02 00 00 03 02 80 05 00 00 00 00 01 01 00 00 D5 41 [00] FF 90 00 [校验和] 03

其中，响应数据 = FF

注：*从存储地址 08 开始更新标签的一个以上的存储内容（块 1：字节-0）。*

HOST -> 02 6F 0D 00 00 00 00 01 00 00 00 FF 00 00 00 08 D4 40 01 58 08 02 AA BB [校验和] 03

RDR -> 02 00 00 03 02 80 06 00 00 00 00 01 01 00 00 D5 41 [00] 90 00 [校验和] 03

其中， 命令 = 58; 起始存储地址 = 08;

写内容的数量 = 02；

存储内容 = AA，BB；

步骤 5. 取消选择标签。

HOST -> 02 6F 08 00 00 00 00 01 00 00 00 FF 00 00 00 03 D4 44 01 [校验和] 03

RDR -> 02 00 00 03 02 80 05 00 00 00 00 01 01 00 00 D5 45 [00] 90 00 [校验和] 03

附录A. Topaz

EEPROM Memory Map										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	0	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	UID-6		Locked
Data	1	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	2	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	3	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	4	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	5	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	6	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes
Data	7	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Data	8	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Data	9	Data64	Data65	Data66	Data67	Data68	Data69	Data70	Data71	Yes
Data	A	Data72	Data73	Data74	Data75	Data76	Data77	Data78	Data79	Yes
Data	B	Data80	Data81	Data82	Data83	Data84	Data85	Data86	Data87	Yes
Data	C	Data88	Data89	Data90	Data91	Data92	Data93	Data94	Data95	Yes
Reserved	D									
Lock/Reserved	E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	

Reserved for internal use
User Block Lock & Status
OTP bits

附录B. Topaz512

EEPROM Memory Map (Segment0)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	00	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	25_h		Locked
Data	01	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	02	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	03	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	04	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	05	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	06	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes
Data	07	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Data	08	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Data	09	Data64	Data65	Data66	Data67	Data68	Data69	Data70	Data71	Yes
Data	0A	Data72	Data73	Data74	Data75	Data76	Data77	Data78	Data79	Yes
Data	0B	Data80	Data81	Data82	Data83	Data84	Data85	Data86	Data87	Yes
Data	0C	Data88	Data89	Data90	Data91	Data92	Data93	Data94	Data95	Yes
Reserved	0D									N/A
Lock/OTP	0E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	N/A
Lock/OTP	0F	OTP-6	OTP-7	LOCK-2	LOCK-3	LOCK-4	LOCK-5	LOCK-6	LOCK-7	N/A

EEPROM Memory Map (Segment1)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	10	Data96	Data97	Data98	Data99	Data100	Data101	Data102	Data103	Yes
Data	11	Data104	Data105	Data106	Data107	Data108	Data109	Data110	Data111	Yes
Data	12	Data112	Data113	Data114	Data115	Data116	Data117	Data118	Data119	Yes
Data	13	Data120	Data121	Data122	Data123	Data124	Data125	Data126	Data127	Yes
Data	14	Data128	Data129	Data130	Data131	Data132	Data133	Data134	Data135	Yes
Data	15	Data136	Data137	Data138	Data139	Data140	Data141	Data142	Data143	Yes
Data	16	Data144	Data145	Data146	Data147	Data148	Data149	Data150	Data151	Yes
Data	17	Data152	Data153	Data154	Data155	Data156	Data157	Data158	Data159	Yes
Data	18	Data160	Data161	Data162	Data163	Data164	Data165	Data166	Data167	Yes
Data	19	Data168	Data169	Data170	Data171	Data172	Data173	Data174	Data175	Yes
Data	1A	Data176	Data177	Data178	Data179	Data180	Data181	Data182	Data183	Yes
Data	1B	Data184	Data185	Data186	Data187	Data188	Data189	Data190	Data191	Yes
Data	1C	Data192	Data193	Data194	Data195	Data196	Data197	Data198	Data199	Yes
Data	1D	Data200	Data201	Data202	Data203	Data204	Data205	Data206	Data207	Yes
Data	1E	Data208	Data209	Data210	Data211	Data212	Data213	Data214	Data215	Yes
Data	1F	Data216	Data217	Data218	Data219	Data220	Data221	Data222	Data223	Yes

EEPROM Memory Map (Segment2)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	20	Data224	Data225	Data226	Data227	Data228	Data229	Data230	Data231	Yes
Data	21	Data232	Data233	Data234	Data235	Data236	Data237	Data238	Data239	Yes
Data	22	Data240	Data241	Data242	Data243	Data244	Data245	Data246	Data247	Yes
Data	23	Data248	Data249	Data250	Data251	Data252	Data253	Data254	Data255	Yes
Data	24	Data256	Data257	Data258	Data259	Data260	Data261	Data262	Data263	Yes
Data	25	Data264	Data265	Data266	Data267	Data268	Data269	Data270	Data271	Yes
Data	26	Data272	Data273	Data274	Data275	Data276	Data277	Data278	Data279	Yes
Data	27	Data280	Data281	Data282	Data283	Data284	Data285	Data286	Data287	Yes
Data	28	Data288	Data289	Data290	Data291	Data292	Data293	Data294	Data295	Yes
Data	29	Data296	Data297	Data298	Data299	Data300	Data301	Data302	Data303	Yes
Data	2A	Data304	Data305	Data306	Data307	Data308	Data309	Data310	Data311	Yes
Data	2B	Data312	Data313	Data314	Data315	Data316	Data317	Data318	Data319	Yes
Data	2C	Data320	Data321	Data322	Data323	Data324	Data325	Data326	Data327	Yes
Data	2D	Data328	Data329	Data330	Data331	Data332	Data333	Data334	Data335	Yes
Data	2E	Data336	Data337	Data338	Data339	Data340	Data341	Data342	Data343	Yes
Data	2F	Data344	Data345	Data346	Data347	Data348	Data349	Data350	Data351	Yes

EEPROM Memory Map (Segment3)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	30	Data352	Data353	Data354	Data355	Data356	Data357	Data358	Data359	Yes
Data	31	Data360	Data361	Data362	Data363	Data364	Data365	Data366	Data367	Yes
Data	32	Data368	Data369	Data370	Data371	Data372	Data373	Data374	Data375	Yes
Data	33	Data376	Data377	Data378	Data379	Data380	Data381	Data382	Data383	Yes
Data	34	Data384	Data385	Data386	Data387	Data388	Data389	Data390	Data391	Yes
Data	35	Data392	Data393	Data394	Data395	Data396	Data397	Data398	Data399	Yes
Data	36	Data400	Data401	Data402	Data403	Data404	Data405	Data406	Data407	Yes
Data	37	Data408	Data409	Data410	Data411	Data412	Data413	Data414	Data415	Yes
Data	38	Data416	Data417	Data418	Data419	Data420	Data421	Data422	Data423	Yes
Data	39	Data424	Data425	Data426	Data427	Data428	Data429	Data430	Data431	Yes
Data	3A	Data432	Data433	Data434	Data435	Data436	Data437	Data438	Data439	Yes
Data	3B	Data440	Data441	Data442	Data443	Data444	Data445	Data446	Data447	Yes
Data	3C	Data448	Data449	Data450	Data451	Data452	Data453	Data454	Data455	Yes
Data	3D	Data456	Data457	Data458	Data459	Data460	Data461	Data462	Data463	Yes
Data	3E	Data464	Data465	Data466	Data467	Data468	Data469	Data470	Data471	Yes
Data	3F	Data472	Data473	Data474	Data475	Data476	Data477	Data478	Data479	Yes

Reserved for internal use
User Block Lock & Status
OTP bits

附录C. Jewel64

EEPROM Memory Map (Segment0)										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	0	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	25_h		Locked
Data	1	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	2	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	3	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	4	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	5	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	6	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes
Data	7	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Reserved	8	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Reserved	9	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Reserved	A	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Reserved	B	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Data	C	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Reserved	D	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	N/A
Lock/OTP	E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	N/A

Reserved for internal use
User Block Lock & Status
OTP bits

附录D. Jewel96

EEPROM Memory Map										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	0	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	UID-6		Locked
Data	1	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	2	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	3	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	4	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	5	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	6	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes
Data	7	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Data	8	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Data	9	Data64	Data65	Data66	Data67	Data68	Data69	Data70	Data71	Yes
Data	A	Data72	Data73	Data74	Data75	Data76	Data77	Data78	Data79	Yes
Data	B	Data80	Data81	Data82	Data83	Data84	Data85	Data86	Data87	Yes
Data	C	Data88	Data89	Data90	Data91	Data92	Data93	Data94	Data95	Yes
Reserved	D									
Lock/Reserved	E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	

Reserved for internal use
User Block Lock & Status
OTP bits

附录E. ACR122 错误代码

错误代码	错误
00h	没有错误。
01h	超时，目标无响应。
02h	非接触 UART 检测到 CRC 错误。
03h	非接触 UART 检测到奇偶校验错误。
04h	在 MIFARE 防冲突/选择操作中，检测到错误的位计数。
05h	MIFARE 卡操作过程中出现帧错误。
06h	以 106 kbps 速率进行逐位补防冲突的过程中检测到异常的位冲突。
07h	通信缓冲区的大小不足。
08h	非接触 UART 检测到 RF 缓冲区溢出（寄存器 CL_ERROR 的 BufferOvfl 位）。
0Ah	在主动通信模式下，对应方没有及时开启 RF 磁场（定义见 NFCIP-1 标准）。
0Bh	RF 协议错误（cf. 参考[4]，CL_ERROR 寄存器的说明）。
0Dh	温度错误：内部温度传感器检测到过热，因此自动关闭了天线的驱动。
0Eh	内部缓冲区溢出
10h	参数无效（范围，结构等）
12h	DEP 协议：在目标模式下配置的芯片不支持从发起者收到的命令（收到的命令不是下列之一：ATR_REQ，WUP_REQ，PSL_REQ，DEP_REQ，DSL_REQ，RLS_REQ，ref. [1]）.
13h	DEP 协议/MIFARE/ISO/IEC 14443-4：数据格式不符合规范。根据采用的 RF 协议，可能是： RF 接收帧的长度错误 PCB 或 PFB 值不正确 无效的或意外的 RF 接收帧 NAD 或 DID 不一致。
14h	MIFARE：认证错误。
23h	ISO/IEC 14443-3：UID 检查字节错误。
25h	DEP 协议：无效的设备状态，系统所处的状态不允许执行该操作。
26h	在此配置下不允许执行操作（主机控制器接口）。
27h	当前的芯片状态导致命令不能被接收（发起方 vs.目标，未知的目标号，目标状态不佳，等等）。
29h	配置为目标的芯片是由其发起方发布的。
2Ah	仅限 ISO/IEC 14443-3B：卡片的 ID 号不匹配，意味着预期的卡片已经被调换。
2Bh	仅限 ISO/IEC 14443-3B：先前激活的卡片消失了。
2Ch	NFCID3 发起方和 NFCID3 目标方在 DEP 212/424 kbps 被动模式下不匹配。

错误代码	错误
2Dh	检测到过流事件。
2Eh	DEP 结构中缺少 NAD。