Advanced Card Systems Ltd.
Card & Reader Technologies

AET62 指纹非接触式智能卡读写器

参考手册 V1.01

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

目录

图目录

表目录

1.0. 简介

AET62 是一款复合设备,包括 ACS 的 ACR122U NFC 读写器内核和 UPEK 的滑动式指纹传感器。NFC 非接触式读写器和指纹传感器既能各自单独使用，也可结合使用提高应用安全级别。AET62 的系统结构如下图所示：

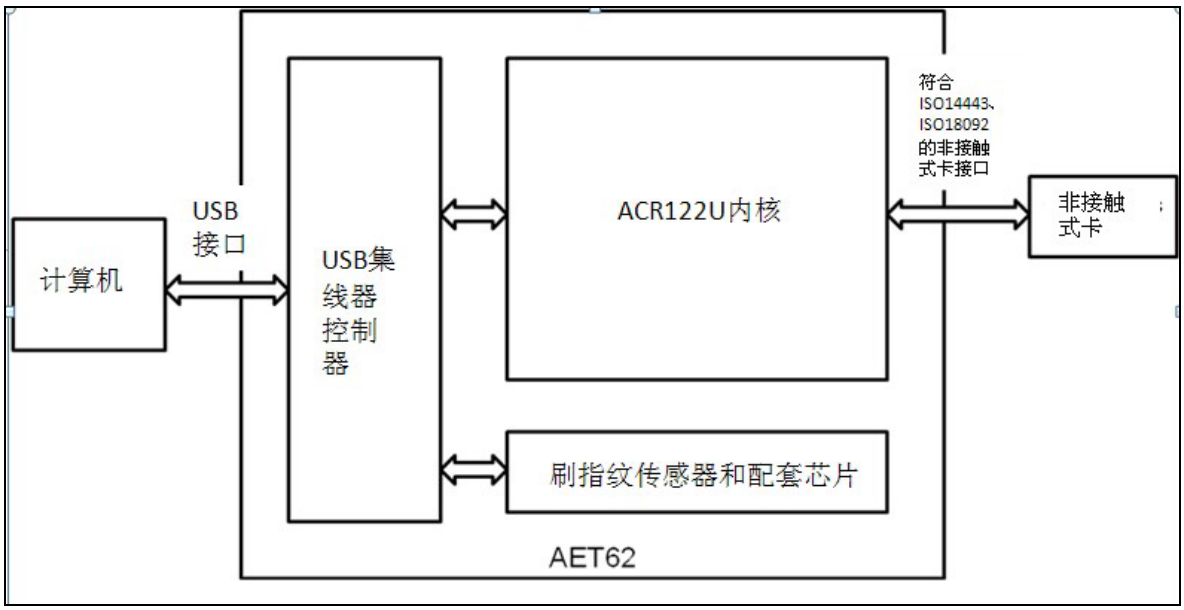


图1 :AET62 系统框图

本手册描述了 AET62 非接触式智能卡读写器模块（基于 ACR122U 内核）的结构和接口。如需了解指纹模块的结构和编程接口，请参考 AET62 应用编程接口文档。

2.0. AET62 非接触式智能卡读写器

AET62 是一款连机非接触式智能卡读写器。它可以读写 ISO 14443-4 A 类和 B 类卡、MIFARE®卡、ISO 18092 或 NFC 卡、以及 FeliCa 标签。符合 PCSC 标准，可兼容现有的 PCSC 应用。另外它采用了标准的 Microsoft CCID 驱动来简化驱动安装程序。

作为非接触式标签与个人电脑的中介设备，AET62 通过 USB 端口连接计算机，并执行计算机命令——包括与非接触标签通信的命令，及控制外围设备（例，双色 LED）的命令。

AET62 用 PCSC APDU 命令操作符合 PCSC 规范的非接触式标签，同时它使用私有 APDU 向 ISO18092 标签发送命令及控制外围设备。本文将介绍如何在智能卡系统中应用 AET62。

2.1. USB 接口

AET62 通过符合 USB 1.1 规范的 USB 端口与计算机建立连接，它支持 USB 全速模式，速率为 12 Mbps。

引脚	信号	功能
1	V_{BUS}	为读写器提供+5 V 的电源（最大 200 mA，常规 100 mA）
2	D-	AET62 和 PC 间以微分信号传输数据。
3	D+	AET62 和 PC 间以微分信号传输数据。
4	GND	参考电压等级

表1： USB 接口

3.0. 功能实现

3.1. AET62 通信流程图

AET62 采用标准的微软 CCID 和 PCSC 驱动，可以直接使用 Windows 操作系统自带的驱动，无需另行安装 ACS 驱动。如需使用 AET62 NFC 读写器的全部功能,请修改计算机注册表的设置。更多细节请参看 **Appendix A** – AET62 PCSC Escape Command 。

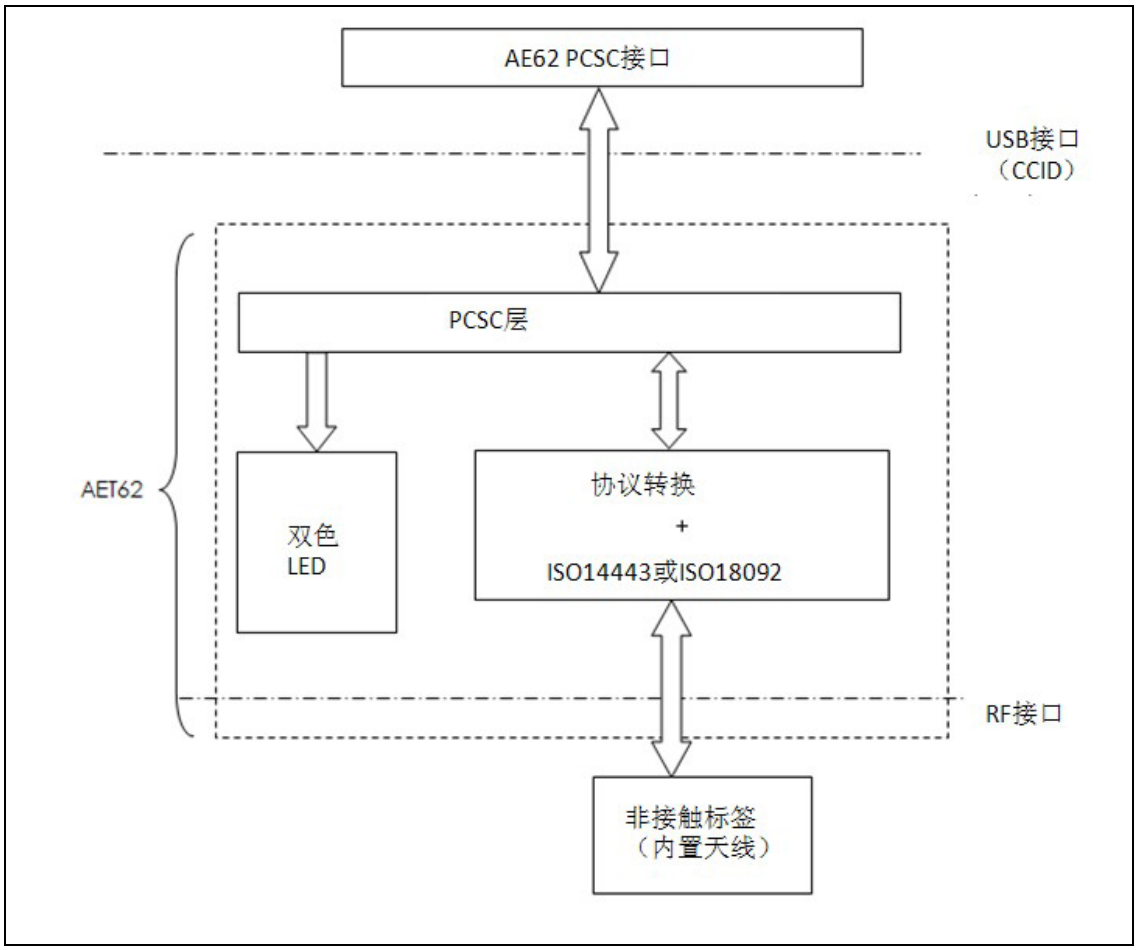


图2 :AET62 通信流程图

3.2. 智能卡读写器接口概述

单击“设备管理器”，找到“AET62 PICC Interface”。设备采用了标准的微软 USB CCID 驱动。

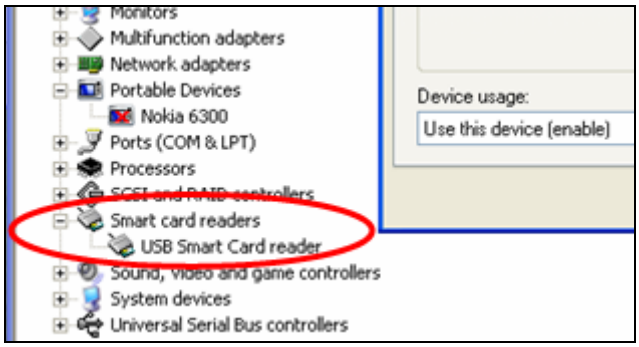


图3 ： 资源管理器中的智能卡读写器接口

4.0. PICC 接口描述

4.1. ATR 的生成

读写器检测到 PICC 后，会发送 ATR 至 PC/SC 驱动来识别 PICC。

4.1.1. ATR 信息格式（适用于 ISO 14443-3 PICC）

字节	值	标记	说明
0	3Bh	初始头部	-
1	8Nh	T0	高半字节 8 表示：后续不存在 TA1、TB1 和 TC1，只存在 TD1。 低半字节 N 表示历史字符的个数（HistByte 0 - HistByte N-1)
2	80h	TD1	高半字节 8 表示：后续不存在 TA2、TB2 和 TC2，只存在 TD2。 低半字节 0 表示协议类型为 T=0
3	01h	TD2	高半字节 0 表示后续不存在 TA3、TB3、TC3 和 TD3。 低半字节 1 表示协议类型为 T=1
4 至 3+N	80h	T1	类别指示字节，80 表示在可选的 COMPACT-TLV 数据对象中或许存在一个状态标识符
	4Fh	Tk	应用标识符存在标识
	0Ch		长度
	RID		注册应用供应商标识(RID) # A0 00 00 03 06h
	SS		标准字节
	C0 ..C1h		卡片名称字节
	00 00 00 00h	RFU	RFU # 00 00 00 00h
4+N	UUh	TCK	T0 至 Tk 的所有字节按位异或

表2： ATR 信息格式（适用于 ISO 14443-3 PICC）

例：

MIFARE 1K 卡的 ATR = {3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6Ah}

ATR											
初始头部	T0	TD1	TD2	T1	Tk	长度	RID	标准	卡片名称	RFU	TCK
3Bh	8Fh	80h	01h	80h	4Fh	0Ch	A0 00 00 03 06h	03h	00 01h	00 00 00 00h	6Ah

其中：

长度 (YY) = 0Ch

RID = A0 00 00 03 06h (PC/SC 工作组)

标准（SS） = 03h（ISO 14443A，第 3 部分）

卡片名称 (C0 ..C1) = [00 01h] (MIFARE 1K)

其中，卡片名称 (C0 ..C1)

00 01h：MIFARE 1K

00 02h：MIFARE 4K

00 03h：MIFARE Ultralight

00 26h：MIFARE Mini

F0 04h：Topaz 和 Jewel

F0 11h：FeliCa 212K

F0 12h：Felica 424K

FFh [SAK]：未定义

4.1.2. ATR 信息格式（适用于 ISO 14443-4 PICC）

字节	值	标记	说明
0	3Bh	初始字符	-
1	8Nh	T0	高半字节 8 表示：后续不存在 TA1、TB1 和 TC1，只存在 TD1。 低半字节 N 表示历史字符的个数（HistByte 0 - HistByte N-1)
2	80h	TD1	高半字节 8 表示：后续不存在 TA2、TB2 和 TC2，只存在 TD2。 低半字节 0 表示协议类型为 T=0
3	01h	TD2	高半字节 0 表示后续不存在 TA3、TB3、TC3 和 TD3。 低半字节 1 表示协议类型为 T=1
4 至 3 + N	XXh	T1	历史字节： ISO 14443A： 来自 ATS 应答的历史字节。请参阅 ISO14443-4 的规定。 ISO 14443B： 来自 ATTRIB 应答（ATQB）的上层响应。参考 ISO 14443-3 标准。
	XXh XX XXh	Tk	
4+N	UUh	TCK	T0 至 Tk 的所有字节按位异或

我们以 DESFire 卡的 ATR 为例：

DESFire (ATR) = 3B 86 80 01 06 75 77 81 02 80 00h

ATR						
初始头部	T0	TD1	TD2	ATS		
				T1	Tk	TCK
3Bh	86h	80h	01h	06h	75 77 81 02 80h	00h

此 ATR 包含六个字节的 ATS：[06 75 77 81 02 80h]

注：*使用 APDU"FF CA 01 00 00h"来区分是符合 ISO 14443A-4 的 PICC 还是符合 ISO 14443B-4 的 PICC，并且如果有的话，取回完整的 ATS。符合 ISO 14443A-3 或 ISO 14443B-3/4 类的 PICC 会返回 ATS。*

再以 ST19XRC8E 的 ATR 为例：

ST19XRC8E (ATR) **=** 3B 8C 80 01 50 12 23 45 56 12 53 54 4E 33 81 C3 55h

ATR						
初始头部	T0	TD1	TD2	ATQB		
				T1	Tk	TCK
3Bh	86h	80h	01h	50h	12 23 45 56 12 53 54 4E 33 81 C3h	55h

由于该卡片符合ISO 14443 B类卡的规定，响应为ATQB，即**50 12 23 45 56 12 53 54 4E 33 81 C3h**，12字节长，不带CRC-B。

注：*更多细节，请参阅 ISO 7816、ISO 14443 和 PC/SC 标准。*

5.0. PICC 常用命令

5.1. 获取数据（Get Data）

该命令用于返回“连接的卡片”的序列号或 ATS。

Get UID 的 APDU 结构（5 个字节）

命令	CLA	INS	P1	P2	Le
Get Data	FFh	CAh	00h 01h	00h	00h (全长)

若 P1 = 00h，响应格式为获取 UID（UID + 2 个字节）

响应	响应数据域					
结果	UID (LSB)	-	-	UID (MSB)	SW1	SW2

若 P1 = 01h，则获取 ISO 14443 A 类卡的 ATS（ATS + 2 个字节）

响应	响应数据域		
结果	ATS	SW1	SW2

响应状态码

结果	SW1 SW2	含义
成功	90 00h	操作成功完成。
错误	63 00h	操作失败。
错误	6A 81h	不支持此功能。

例：

1. 获取“已经建立连接的 PICC”的序列号:
   UINT8 GET_UID[5]={FFh, CAh, 00h, 00h, 04h};
2. 获取“已经建立连接的 ISO 14443-A PICC”的 ATS:
   UINT8 GET_ATS[5]={FFh, CAh, 01h, 00h, 04h};

6.0. MIFARE Classic 存储卡的 PICC 命令（T=CL 模拟）

6.1. 加载认证密钥（Load Authentication Keys）

该命令用于向读写器加载认证密钥。该认证密钥用于验证 MIFARE 1K/4K 存储卡的特定扇区。读写器提供了两种认证密钥位置，分别是易失密钥位置和非易失密钥位置。

Load Authentication Keys 的 APDU 结构（11 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Load Authentication Keys	FFh	82h	密钥结构	密钥号	06h	密钥（6 个字节）

其中：

密钥结构 1 个字节。

00h = 密钥被载入读写器的易失存储器。

其它 = 保留。

密钥号 1 个字节。

00h ~ 01h = 密钥位置。一旦读写器与电脑断开连接，密钥就会消失。

密钥 6 个字节。

载入读写器的密钥值，例：{FF FF FF FF FF FFh}

Load Authentication Keys 的响应结构（2 个字节）

响应	响应数据域	
结果	SW1	SW2

响应状态码

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：

向密钥位置 00h 加载密钥{FF FF FF FF FF FFh}。

APDU = {FF 82 00 00h 06 FF FF FF FF FF FFh}

6.2. 认证（Authentication）

该命令使用存储在读写器内的密钥来验证 MIFARE 1K/4K 卡（PICC）。涉及两种认证密钥：TYPE_A 和 TYPE_B。

Load Authentication Keys 的 APDU 结构（6 个字节）[弃用]

命令	CLA	INS	P1	P2	P3	命令数据域
Authentication	FFh	88h	00h	块号	密钥类型	密钥号

Load Authentication Keys 的 APDU 结构（10 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Authentication	FFh	86h	00h	00h	05h	认证数据字节

认证数据字节（5 个字节）

字节 1	字节 2	字节 3	字节 4	字节 5
版本号 01h	00h	块号	密钥类型	密钥号

其中：

块号 1 个字节。表示待验证的存储块。

密钥类型 1 个字节。

60h = 该密钥被用作 TYPE A 密钥进行验证

61h = 该密钥被用作 TYPE B 密钥进行验证

密钥号 1 个字节。

00h ~ 01h = 密钥位置。

注：*MIFARE 1K 卡的内存划分为 16 个扇区，每个扇区包含 4 个连续的块。例：扇区 00h 包含块{00h、01h、02h 和 03h}；扇区 01h 包含块{04h、05h、06h 和 07h}；最后一个扇区 0Fh 包含块{3Ch、3Dh、3Eh 和 3Fh}。*

验证通过后，读取同一扇区内的其他块不需要再次进行验证。详情请参考 MIFARE 1K/4K 卡标准。

Load Authentication Keys 的响应结构（2 个字节）

响应	响应数据域	
结果	SW1	SW2

响应状态码

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

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

1K 字节

表3： MIFARE 1K 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

2K 字节

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4： MIFARE 4K 内存结构

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	内部	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5： MIFARE Ultralight 卡的内存结构

例：

1. 要使用{TYPE A，密钥号 00h}验证块 04h。PC/SC V2.01，弃用
   APDU = {FF 88 00 04 60 00h};
2. 要使用{TYPE A，密钥号 00h}验证块 04h。PC/SC V2.07
   APDU = {FF 86 00 00 05 01 00 04 60 00h}

注：MIFARE Ultralight 不需要进行验证，其内存可以自由访问。

6.3. 读取二进制块（Read Binary Blocks）

该命令用于获取 PICC 卡片中的多个“数据块”。执行该命令前必须先对数据块/尾部块进行验证。

Read Binary 的 APDU 结构（5 个字节）

命令	CLA	INS	P1	P2	Le
Read Binary Blocks	FFh	B0h	00h	块号	待读取的字节数

其中：

块号 1 个字节。待访问的块。

待读取的字节数 1 个字节。最大值为 16 个字节。

Read Binary Block 的响应结构（N + 2 个字节）

响应	响应数据域		
结果	0 <= N <= 16	SW1	SW2

响应状态码

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. 从二进制块 04h 中读取 16 个字节（MIFARE 1K 或 4K）
   APDU = {FF B0 00 04 10h}
2. 从二进制页 04h 中读取 4 个字节（MIFARE Ultralight）
   APDU = {FF B0 00 04 04h}
3. 从二进制页 04h 开始读取 16 个字节（MIFARE Ultralight）（读取页 4, 5, 6 和 7）
   APDU = {FF B0 00 04 10h}

6.4. 更新二进制块（Update Binary Blocks）

该命令用于向 PICC 写入多个“数据块”。执行该命令前必须先验证数据块/尾部块。

Update Binary 的 APDU 结构（4 或 16 + 5 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Update Binary Blocks	FFh	D6h	00h	块号	待更新的字节数	块数据 MIFARE Ultralight：4 个字节；或 MIFARE 1K/4K：16 个字节

其中：

块号 1 个字节。待更新的起始块

待更新的字节数 1 个字节。

MIFARE 1K/4K 的待更新字节数为 16 个字节

MIFARE Ultralight 的待更新字节数为 4 个字节

块数据 （4 或 16 个字节）：待写入二进制块的数据。

响应状态码

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. 将 MIFARE 1K/4K 卡中的二进制块 04h 的数据更新为{00 01 ..0Fh}

APDU = {FF D6 00 04 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0Fh}

2. 将 MIFARE Ultralight 中二进制块 04h 的数据更新为{00 01 02 03}

APDU = {FF D6 00 04 04 00 01 02 03h}

6.5. 与值块相关的的命令

数据块可以用作值块来执行基于数值的应用。

6.5.1. 值块操作（Value Block Operation）

该命令用于处理基于数值的交易，例：增加值块的值等。

Value Block Operation 的 APDU 结构（10 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域	
Value Block Operation	FFh	D7h	00h	块号	05h	VB_OP	VB_Value（4 个字节）{MSB ..LSB}

其中：

块号 1 个字节。待操作的值块

VB_OP 1 个字节。

00h = 将 VB_Value 存入该块，然后该块将变为值块。

01h = 使值块的值增加 VB_Value。该命令仅用于操作值块。

02h = 使值块的值减少 VB_Value。该命令仅用于操作值块。

VB_Value（4 个字节）用于算数运算的数值，是一个有符号长整数（4 个字节）。

例 1：Decimal –4 = {FFh, FFh, FFh, FCh}

VB_Value			
MSB			LSB
FFh	FFh	FFh	FCh

例 2：Decimal 1 = {00h, 00h, 00h, 01h}

VB_Value			
MSB			LSB
00h	00h	00h	01h

Value Block Operation 的响应结构（2 个字节）

响应	响应数据域	
结果	SW1	SW2

响应状态码

结果	SW1 SW2	含义
成功	90 00h	操作成功完成。

结果	SW1 SW2	含义
错误	63 00h	操作失败。

6.5.2. 读取值块（Read Value Block）

该命令用于获取值块中的数值，该命令仅用于操作值块。

Value Block Operation 的 APDU 结构（5 个字节）

命令	CLA	INS	P1	P2	Le
Read Value Block	FFh	B1h	00h	块号	04h

其中：

块号 1 个字节。待访问的值块

Read Value Block 的响应结构（4 + 2 个字节）

响应	响应数据域		
结果	值 {MSB ..LSB}	SW1	SW2

其中：

值（4 个字节）.卡片返回的数值，是一个有符号长整数（4 个字节）。

例 1：Decimal –4 = {FFh, FFh, FFh, FCh}

值			
MSB			LSB
FFh	FFh	FFh	FCh

例 2：Decimal 1 = {00h, 00h, 00h, 01h}

值			
MSB			LSB
00h	00h	00h	01h

响应状态码

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

6.5.3. 恢复值块（Restore Value Block）

该命令用于将一个值块中的数值复制到另外一个值块。

Restore Value Block 的 APDU 结构（7 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域	
Restore Value Block	FFh	D7h	00h	源块号	02h	03h	目标块号

其中：

源块号 1 个字节。源值块中的值会被复制到目标值块。

目标块号 1 个字节。要恢复的值块。源值块和目标值块必须位于同一个扇区。

Restore Value Block 的响应结构（2 个字节）

响应	响应数据域	
结果	SW1	SW2

响应状态码

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. 将数值“1”存入块 05h

   APDU = {FF D7 00 05 05 00 00 00 00 01h}

   应答：90 00h

2. 读取值块 05h

   APDU = {FF B1 00 05 00h}

   应答：00 00 00 01 90 00h [9000h]

3. 将值块 05h 的值复制到值块 06h

   APDU = {FF D7 00 05 02 03 06h}

   应答：90 00h [9000h]

4. 使值块 05h 的值增加“5”

   APDU = {FF D7 00 05 05 01 00 00 00 05h}

   应答：90 00h [9000h]

7.0. 私有 APDU

私有 APDU 用于以下目的：

- 与不符合 PCSC 规范的标签交换数据。
- 取回或设置读写器的参数。
- 如果已经与标签建立连接，则私有 APDU 可以通过“AET62 PICC Interface”来发送。
- 若还没有标签，则可以使用“直接（Escape）命令”来发送私有 APDU。

7.1. 直接传输（Direct Transmit）

该命令用于发送到标签或读写器的数据包。

Direct Transmit 的命令结构（数据包的长度 + 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	数据包

其中：

Lc 1 个字节。待发送的字节数。
最大值为 255 个字节。

命令数据域 响应。

Direct Transmit 的响应结构

响应	响应数据域
Direct Transmit	响应数据

7.2. 双色 LED 控制（Bi-color LED Control）

该命令用于控制双色 LED 指示灯的状态。

Bi-color LEDs 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 Control	FFh	00h	40h	LED 状态控制	04h	闪烁周期控制

P2 LED 状态控制

Bi-color LEDs Control 的结构（1 个字节）

命令	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

其中：

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	00h

其中：

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。

其中：

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。比如说，如果 T1=1，T2=1，则工作周期 = 50%。工作周期 = T1/(T1 + T2).*

7.3. 获取固件版本号（Get Firmware Version）

该命令用于获取读写器的固件版本号。

命令结构（5 个字节）

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

响应结构（10 个字节）

响应	响应数据域
结果	固件版本号

示例响应 = 41 45 54 36 32 30 33 30 30h = AET620300 (ASCII)

7.4. 获取读写器的 PICC 操作参数（Get PICC Operating Parameter）

该命令用于获取读写器的 PICC 操作参数。

命令结构（5 个字节）

命令	CLA	INS	P1	P2	Le
Get Response	FFh	00h	50h	00h	00h

响应结构（1 个字节）

响应	响应数据域
结果	PICC 操作参数

7.5. 设置 PICC 操作参数（Set the PICC Operating Parameter）

该命令用于设置读写器的 PICC 操作参数。

命令结构（5 个字节）

命令	CLA	INS	P1	P2	Le
Get Response	FFh	00h	51h	新的 PICC 操作参数	00h

响应结构（1 个字节）

响应	响应数据域
结果	PICC 操作参数

位	参数	说明	选项
7	自动 PICC 轮询	启用 PICC 轮询。	1 = 启用 0 = 停用
6	自动 ATS 生成	每次激活 ISO 14443-4 A 类标签都发送 ATS 请求	1 = 启用 0 = 停用
5	轮询时间间隔	设置连续 PICC 轮询之间的时间间隔	1 = 250 ms 0 = 500 ms
4	FeliCa 424K	PICC 轮询中待检测标签的类别	1 = 检测 0 = 跳过
3	FeliCa 212K		1 = 检测 0 = 跳过
2	Topaz		1 = 检测 0 = 跳过
1	ISO 14443 B 类		1 = 检测 0 = 跳过
0	ISO 14443 A 类 #要检测 MIFARE 标签，必须首先禁用 ATS 自动生成。		1 = 检测 0 = 跳过

PICC 操作参数默认值 = FFh。

8.0. 非接触式应用的基本流程

步骤 0.启动应用程序，读写器会不断地进行 PICC 轮询和标签扫描。

一旦发现并检测到标签，相应的 ATR 会被发送到 PC。您必须确保已经设置了 PCSC Escape 命令。更多细节请参看 **Appendix A** - AET62 PCSC Escape Command 。

步骤 1.首先要连接“AET62 PICC Interface”。

步骤 2.发送 APDU 命令来访问 PICC。

:

:

步骤 N. 断开“AET62 PICC Interface”的连接，关闭应用程序。

注：

1. *您可以关掉天线来节省电源。*
   - *关闭天线电源：FF 00 00 00 04 D4 32 01 00*
   - *开启天线电源：FF 00 00 00 04 D4 32 01 01*
2. *处理标准的 APDU 和非标准的 APDU。*
   - *采用标准 APDU 格式的 PICC：ISO14443-4 A 类和 B 类、MIFARE .. 等等*
   - *采用非标准 APDU 格式的 PICC：FeliCa, Topaz .. 等等*

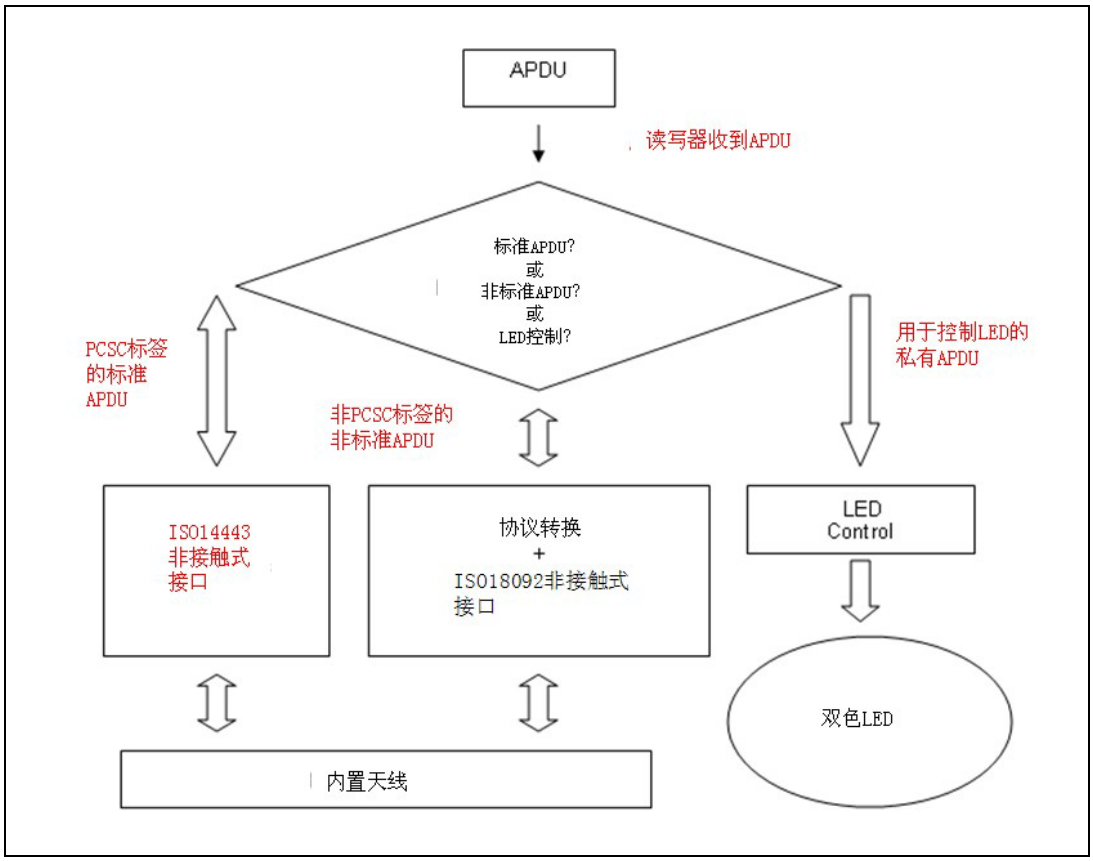


图4：非接触式应用的基本流程

AET62 的 PICC 接口采用了 ISO7816 T=1 协议。

- PC → 读写器：向读写器发送 APDU。
- 读写器 → PC：返回响应数据。

8.1. 访问符合 PCSC 标准的标签（ISO 14443-4）

基本上，所有符合 ISO14443-4 标准的卡片（PICC 卡）都可以理解 ISO 7816-4 规定的 APDUs。AET62 读写器与符合 ISO 14443-4 标准的卡片进行通信时，只需要对 ISO 7816-4 规定的 APDU 和响应进行转换。AET62 会在内部处理 ISO 14443 第 1-4 部分协议。

另外 MIFARE 1K、4K、MINI 和 Ultralight 标签是通过 T=CL 模拟进行支持的，只要将 MIFARE 标签视作标准的 ISO 14443-4 标签即可。更多相关信息，请参阅“MIFARE Classic 存储标签的 PICC 命令”。

ISO 7816-4 规定的 APDU 报文的结构

命令	CLA	INS	P1	P2	Lc	命令数据域	Le
ISO 7816-4 规定的命令					命令数据域的长度		期望返回的响应数据的长度

ISO 7816-4 规定的响应报文的结构（数据 + 2 个字节）

响应	响应数据域		
结果	响应数据	SW1	SW2

通用的 ISO 7816-4 命令的响应状态码

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

典型的操作顺序为：

1. 出示标签，并连接 PICC 界面。
2. 读取/更新标签的存储内容。

步骤 1. 与标签建立连接。

步骤 2. 发送 APDU，取随机数

<< 00 84 00 00 08h。

>> 1A F7 F3 1B CD 2B A9 58 [90 00h]

注：*对于 ISO 14443-4 A 类标签来说，可以通过 APDU“FF CA 00 00 01”来获取 ATS。*

8.2. 访问 DESFire 标签（ISO 14443-4）

DESFire 支持 ISO 7816-4 APDU 包模式和本地模式。一旦 DESFire 标签被激活，发送至 DESFire 标签的第一个 APDU 就会确定“命令的模式”。如果第一个 APDU 采用“本地模式”，则其余的 APDU 都必须是“本地模式”。同样，如果第一个 APDU 采用“ISO 7816-4 APDU 包模式”，则其余的 APUDU 都必须是“ISO 7816-4 APDU 包模式”。

例 1：DESFire ISO 7816-4 APDU 包

从 ISO 14443-4 Type A PICC (DESFire)中读取 8 个字节的随机数

APDU = {90 0A 00 00 01 00 00}

CLA = 90h; INS = 0Ah (DESFire Instruction); P1 = 00h; P2 = 00h

Lc = 01h; Data In = 00h; Le = 00h (Le = 00 for maximum length)

应答：7B 18 92 9D 9A 25 05 21h [$91AFh]

状态码[91 AFh]的定义见 DESFire 标准。详情请参阅 DESFire 标准。

例 2：DESFire 分页链接（ISO 7816 APDU 包模式）

在本例中，应用涉及到"分页链接"。要获取 DESFire 卡的版本号：

步骤 1：发送 APDU {90 60 00 00 00h}来获取第一个数据页。INS=60h

应答：04 01 01 00 02 18 05 91 AFh [$91AFh]

步骤 2：发送 APDU {90 AF 00 00 00h}来获取第二个数据页。INS=AFh

应答：04 01 01 00 06 18 05 91 AFh [$91AFh]

步骤 3：发送 APDU {90 AF 00 00 00h}来获取最后一个数据页。INS=AFh

应答：04 52 5A 19 B2 1B 80 8E 36 54 4D 40 26 04 91 00h [$9100h]

例 3：DESFire 本地命令

若本地 DESFire 命令更易于操作，则我们可以向读写器发送不带 ISO 7816 包的本地 DESFire 命令。

从 ISO 14443-4 Type A PICC (DESFire)中读取 8 个字节的随机数：

APDU = {0A 00h}

应答：AF 25 9C 65 0C 87 65 1D D7h [$1DD7h]

其中，第一个字节“AF”是 DESFire 卡片返回的状态码。

应用程序可以对[$1DD7]中的数据予以忽略。

例 4：DESFire 分页链接 (本地模式)

在本例中，应用涉及到"分页链接"。

要获得 DESFire 卡的版本号：

步骤 1：发送 APDU {60h} 来获取第一个数据页。INS=60h

应答：AF 04 01 01 00 02 18 05h [$1805h]

步骤 2：发送 APDU {AFh} 来获取第二个数据页。INS=AFh

应答：AF 04 01 01 00 06 18 05h [$1805h]

步骤 3：发送 APDU {AFh} 来获取最后一个数据页。INS=AFh

应答：00 04 52 5A 19 B2 1B 80 8E 36 54 4D 40 26 04h [$2604h]

注：*在 DESFire 本地模式下，如果响应的长度大于 1，则在响应中不会出现状态码[90 00h]。但是如果响应的长度小于 2，则会根据 PCSC 的要求在响应中增加状态码[90 00h]。最短的响应长度为 2。*

8.3. 访问 FeliCa 标签（ISO 18092）

典型的操作顺序为：

1. 出示 FeliCa 标签，并与 PICC 接口建立连接。
2. 读取/更新标签的存储内容。

步骤 1. 与标签建立连接。

The ATR = 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 F0 11 00 00 00 00 8A

其中，

F0 11 = FeliCa 212K

步骤 2. 读取内存块，**不使用私有的 APDU**。

<< 10 06 [8-byte NFC ID] 01 09 01 01 80 00

>> 1D 07 [8-byte NFC ID] 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 [90 00]

OR

步骤 2. 读取内存块，**使用私有的 APDU**。

<< **FF 00 00 00 [13] D4 40 01** 10 06 [8-字节 NFC ID] 01 09 01 01 80 00

其中，

[13h] 是私有数据“**D4 40 01**..80 00”的长度

D4 40 01 是数据交换（Data Exchange）命令

>> **D5 41 00** 1D 07h [8-字节 NFC IDh] 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 [90 00]

其中，**D5 41 00** 是对 Data Exchange 的响应

注意：可以使用 APDU“FF CA 00 00 00”来获取 NFC ID.

详情请参阅 Felica 标准的相关规定。

8.4. 访问 NFC 论坛 1 类标签（ISO 18092）

例：Jewel 和 Topaz 标签。

典型的操作顺序为：

1. 出示 Topaz 标签，并与 PICC 接口建立连接。
2. 读取/更新标签的存储内容。

步骤 1. 与标签建立连接。

ATR = 3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 F0 04 00 00 00 00 9Fh

其中，F0 04 = Topaz

步骤 2. 读取内存地址 **08h (Block 1:Byte-0）**，不使用私有 APDU

<< **01 08h**

>> **18h** [90 00h]

其中，响应数据 = **18h**

OR

步骤 2. 读取内存地址 **08h** (Block 1:**Byte-0**），使用私有 APDU

<< **FF 00 00 00 [05] D4 40 01 01 08h**

其中，

[05h] 是私有 APDU 数据“**D4 40 01** 01 **08h**”的长度

D4 40 01h 是数据交换（Data Exchange）命令

01 **08h** 是要发送给标签的数据。

>> **D5 41 00 18h** [90 00h]

其中，响应数据 = **18h**

注：读取整个标签的存储内容：

<< **00h**

>> 11 48 18 26 ..00 [90 00h]

步骤 3. 将内存地址 **08h**（Block 1:**Byte-0**）**更新为数据 FFh**

<< **53 08 FFh**

>> FFh [90 00h]

其中，响应数据 = FFh

内存地址 = 块号 * 8 + 字节数

例：内存地址 08 (hex) = 1 x 8 + 0 = Block 1:Byte-0 = Data0

例：内存地址 10 (hex) = 2 x 8 + 0 = Block 2:Byte-0 = Data8

HR0	HR1
11_h	xx_h

EEPROM内存结构

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	

仅限内部使用
用户块锁和状态
OTP位

图5：Topaz 内存图

注：*详情请参阅 Jewel 和 Topaz 规范的相关规定。*

8.5. 获取非接触式接口的当前设置

步骤 1. 执行 Get Status Command 命令。

<< FF 00 00 00 02 D4 04h

>> D5 05h [Err] [Field] [NbTg] [Tg] [BrRx] [BrTx] [Type] 80 90 00h

如果天线场内没有标签：

>> D5 05 00 00 00 80 90 00h

[Err] 是一个错误代码，对应于最新检测到的错误。

Field 表示是否存在并检测到外部 RF 磁场，（Field = 01h：是）或（Field = 00h：否)。

[NbTg] 表示目标数。默认值为 1。

[Tg]：逻辑编号

[BrRx]：数据接收的比特率

00h：106 kbps

01h：212 kbps

02h:424 kbps

[BrTx]：数据传输的比特率

00h：106 kbps

01h：212 kbps

02h:424 kbps

[Type]：调制方式

00h：ISO 14443 或 MIFARE

10h:FeliCa

01h：主动式

02h:Innovision Jewel 标签

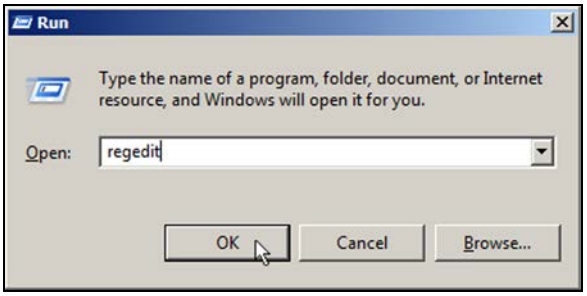
附录A AET62 PCSC 直接命令

1. 选择“**ACS AET62 PICC Interface 0**”。
2. 如果“**AET62 PICC Interface**”已经连接，选择“Shared Mode”；如果“**AET62 PICC Interface**”尚未连接，选择“Direct Mode”。
3. 按下 **Connect** 按钮，在计算机和 AET62 读写器间建立连接。
4. 在命令文本框中输入“3500”。
5. 输入 PCSC Escape 命令，例“**FF 00 48 00 00h**”；然后点击“**Send**”，将命令发送给读写器（获取固件版本号）。
6. 点击 **Disconnect** 来断开连接。
7. 要将 Escape 命令发送或接收到读写器，请按照下列步骤执行。
8. Escape 命令的供应商 IOCTL 定义如下：

   #define IOCTL_CCID_ESCAPE SCARD_CTL_CODE(3500)

下面介绍的是启用 PCSC Escape 命令的详细步骤：

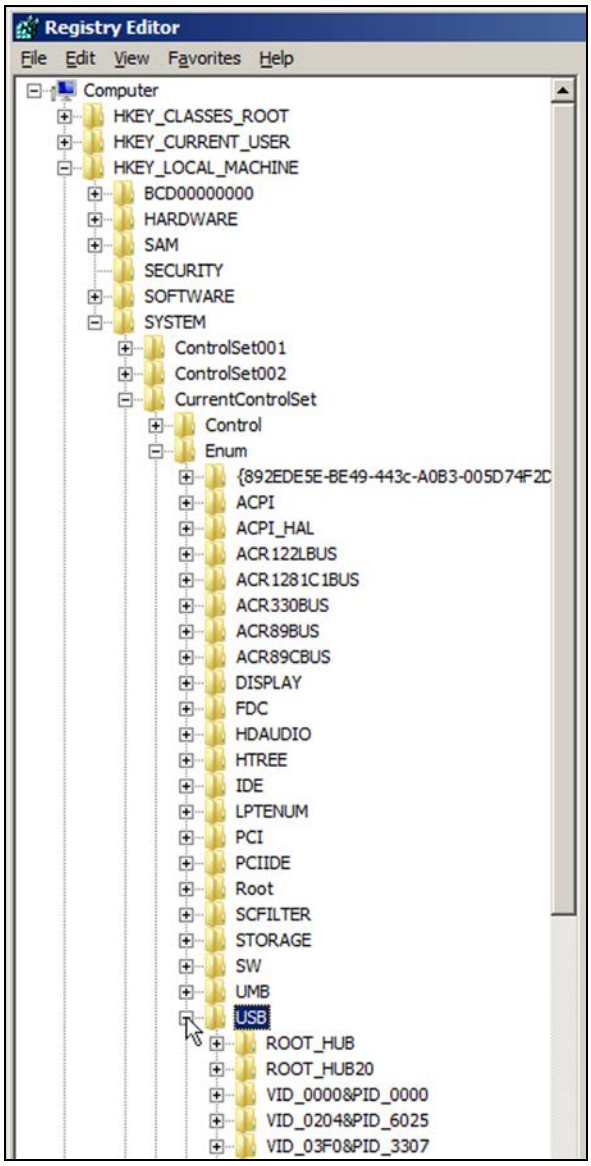
1. 在 Windows 的“Run”命令菜单内执行“regedit”。



2. 在 HKLM\SYSTEM\CCS\Enum\USB\Vid_072F&Pid_90CC\Device Parameters 下添加一个 DWORD“EscapeCommandEnable”。

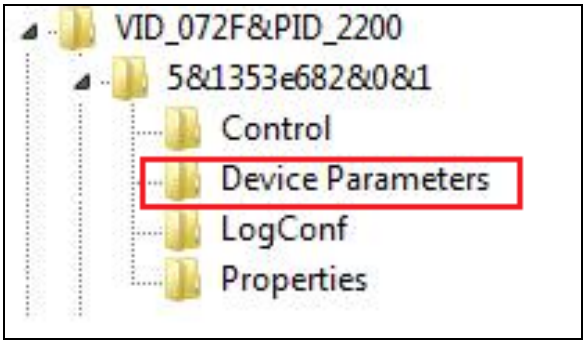
   对于 Vista 系统，路径为：

   Computer\HKEY_LOCAL_MACHINE\SYSTEMS\CurrentControlSet\Enum\USB

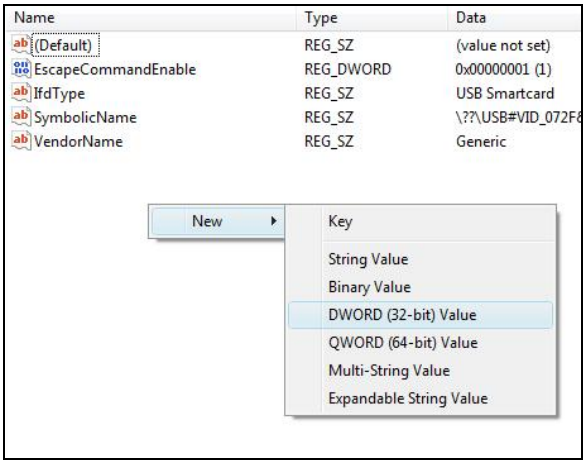


3. 找到：VID_072F&PID_2200

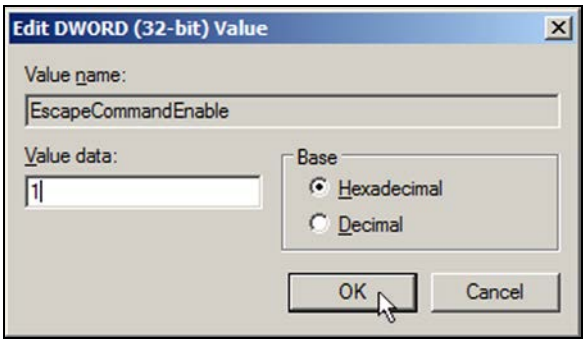
   然后展开该节点。查看 Device Parameters



4. 创建一个 DWORD 项（32 位），将其命名为 EscapeCommandEnable



5. 要修改 EscapeCommandEnable 的值，请双击该项，在 Value data 内输入 1，并将 Base 设为 Hexadecimal。



附录B 符合 ISO 14443 标签的 APDU 命令和响应

假设使用的是一张 ISO 14443-4 Type B 标签。

<< 典型的 APDU 命令和响应流程 >>

PC	读写器	标签
顺序	USB 接口 (12 Mbps)	RF 接口 (13.56 MHz)
1.发送命令.	与非接触相关的命令 → [APDU 命令] 例：[00 84 00 00 08h] (取随机数)	标签特定的命令结构 → [APDU 命令]符合 ISO 14443 定义的结构
2.接收响应。	与非接触相关的响应 ← [APDU 响应] 例：[11 22 33 44 55 66 77 88h] (90 00h)	标签特定的响应结构 ← [APDU 响应]符合 ISO 14443 定义的结构

附录C　　符合 ISO 18092 标签的 APDU 命令和响应

假设使用的是一张 TOPAZ 标签

<< 典型的 APDU 命令和响应流程 >>

PC	读写器	标签
顺序	USB 接口 (12 Mbps)	RF 接口 (13.56 MHz)
1.发送命令.	与非接触相关的命令 →	标签特定的命令结构 →
	[本地命令] 例：[01 08h] (读取内存地址 08h）	[本地命令]符合 ISO 18092 定义的结构
	或	
	私有 APDU 命令 + [本地命令] 例：FF 00 00 00 05 D4 40 01h [01 08h]	
2.接收响应。	与非接触相关的响应 ←	标签特定的响应结构 ←
	[本地响应] 例：00h (90 00h)	例：[本地响应]符合 ISO 18092 定义的结构
	或	
	私有 APDU 响应 + [本地响应] 例：D5 41 00h [00h] (90 00h)	

附录D 错误代码

错误码	错误
00h	没有错误。
01h	超时，目标无应答。
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。

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

附录E　　设置 LED 的示例代码

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：打开红色和绿色 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”

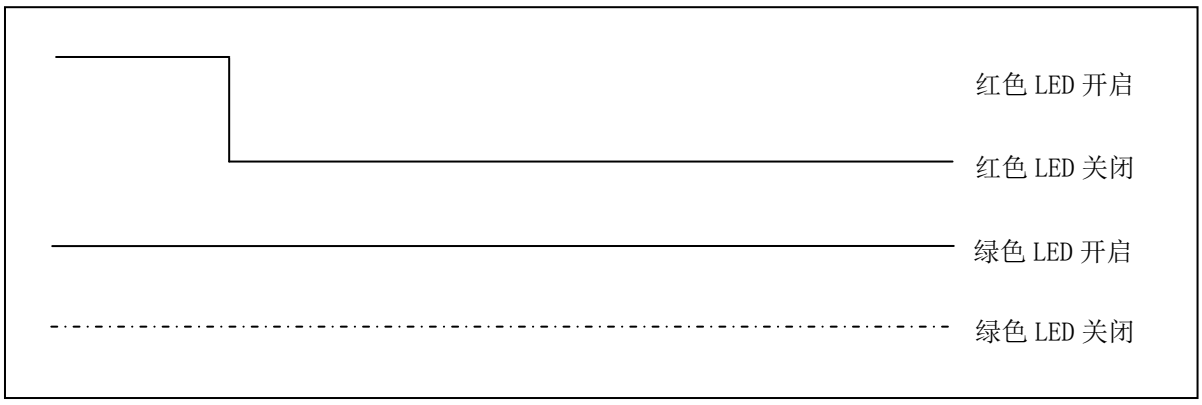
例 3：只关闭红色的 LED，绿色的 LED 保持不变。

// 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”

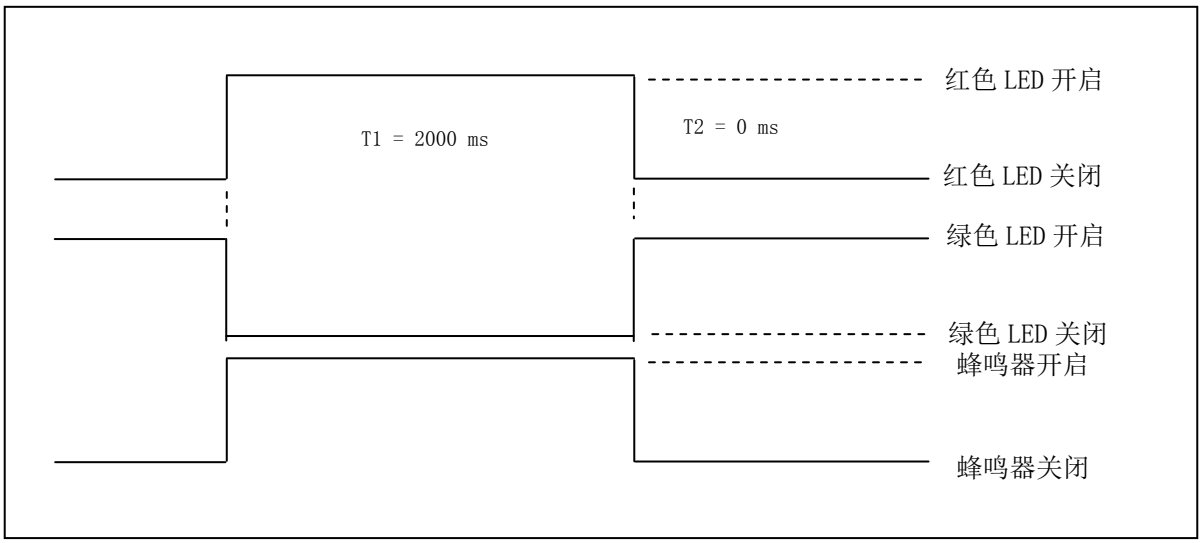
响应 = “90 02”。绿色 LED 保持不变（开启）；红色 LED 关闭，



例 4：将红色的 LED 开启 2 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”

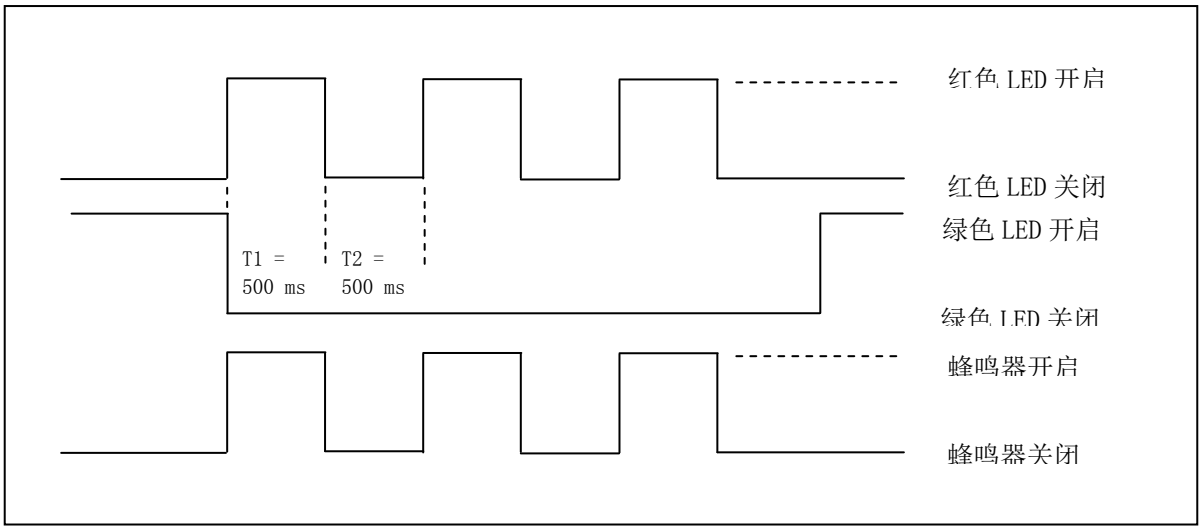
例 5：使红色 LED 以 1Hz 的频率闪烁，然后回到初始状态。

// 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”

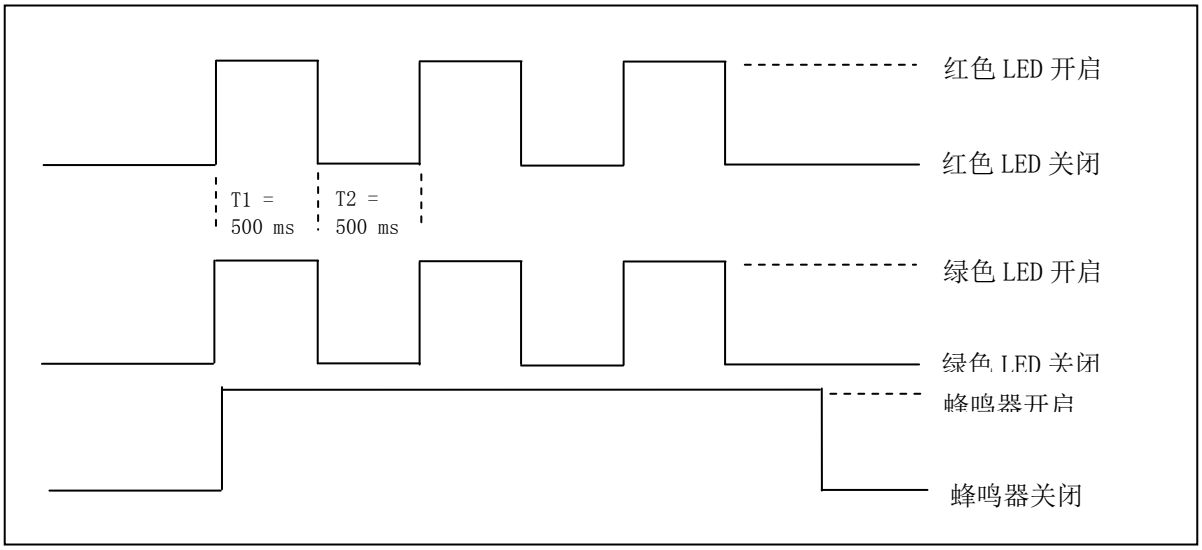
响应 = “90 00”

例 6：使红色 LED 和绿色 LED 均闪烁 3 次，每次 1Hz。

// 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”

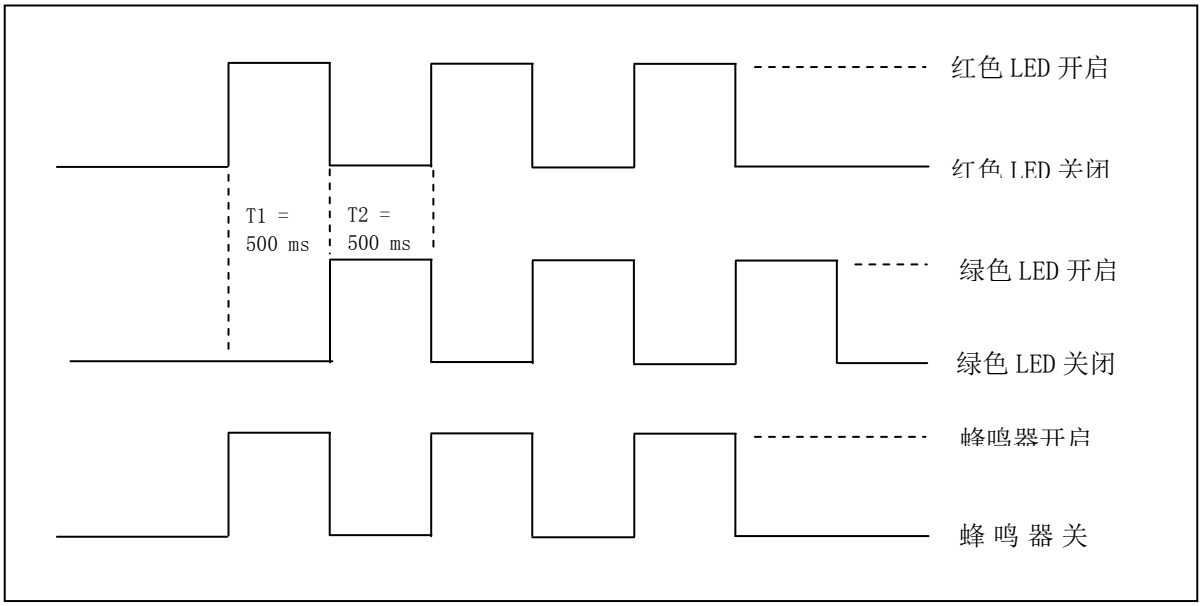
响应 = “90 00”

例 7：使红色 LED 和绿色 LED 轮流闪烁 3 次，频率都是 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 D0 04 05 05 03 01”；响应 = “90 00”