PIOTEC

PIOTEC 非接触读写器驱动接口开发说明书

说明文档

沈阳派尔泰科科技有限公司

文档信息

文档名称	PIOTEC 非接触读写器驱动接口开发说明书
类别	说明文档
编号	PT-READER-WD004-002
文档说明	

修订历史

版本	日期	章节	类型	作者	摘要
V1.0	2017/12/14	全篇	创建	封晓帆	
V1.1	2017/12/16	全篇	修改	修庆国	
V1.2	2018/01/16		增加	封晓帆	增加配置文件位置说明
V1.3	2018/01/19		增加	封晓帆	增加软件接口说明
V1.4	2018/01/23		修改	封晓帆	修改 MIFARE 卡认证函数参数说明
V1.5	2021/05/11		修改	封晓帆	增加 MIFARE 卡读取和写入接口

目录

图片目录

表格目录

第1章 概述

1.1 目的

本文档描述了 PIOTEC 非接触读写器的软件结构与原理，用于指导用户软件开发人员对 PIOTEC 非接触读写器进行二次开发，进而驱动安装在智能卡个性化设备上的PIOTEC 非接触读写器完成卡片的个性化数据写入。

1.2 适用范围

本文档适用于由 PIOTEC 研发并生产的 PT1301N 型号非接触读写器。

1.3 术语表

表 1.1 术语表

序号	术语	简称	说明
1	个性化脚本	脚本	存储符合 ISO14443A/B、ISO15693、MIFARE 和 FELICA 规范的指令文件。

1.4 SDK 目录结构

表 1.2SDK 目录结构

序号	目录名称	说明
1	doc	存放 PIOTEC 读写器驱动接口开发说明书。
2	PTCtlsReaderTest	用户二次开发的 pt_card.dll 程序调试工程。
3	PRE Compiler template	用户二次开发的 PRE 程序编译模板工程。
4	PREDebug	用户二次开发的 PRE 程序调试工程。
5	Reader Lib	用户二次开发的相关库文件。
6	bin	用户二次开发 PTCtlsReaderTest 调试程序目录。

第2章 软件结构

2.1 指令模式软件结构

在指令模式下，用户需针对集成在用户数据管理系统中的组件进行二次开发，如图 2.1 橘黄色部分所示。

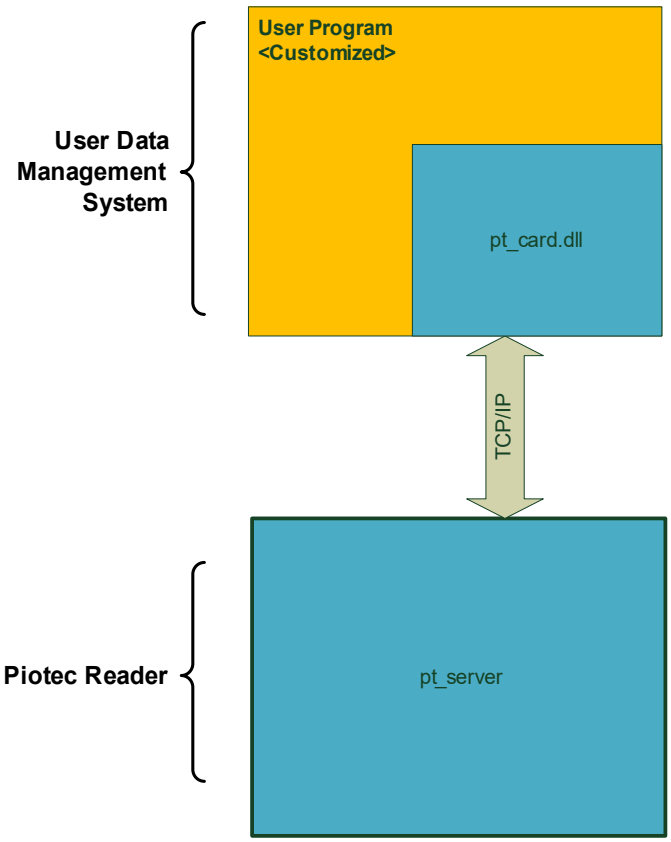

图 2.1 指令模式软件结构图

指令模式下的各系统组件说明如下：

- pt_card.dll ：Piotec 读写器控制模块，通过网络通讯协议与 Piotec 读写器进行指令与数据交互，从而实现对 Piotec 读写器的各类操作。
- pt_server：读写器服务程序，完成与读写器控制模块的网络通讯、读写器内部程序的调用。

2.2 嵌入模式软件结构

在嵌入模式下，用户需针对两个系统组件进行二次开发，其一集成在用户数据管理系统中，其二集成在 Piotec 读写器中，如图 2.2 橘黄色部分所示。

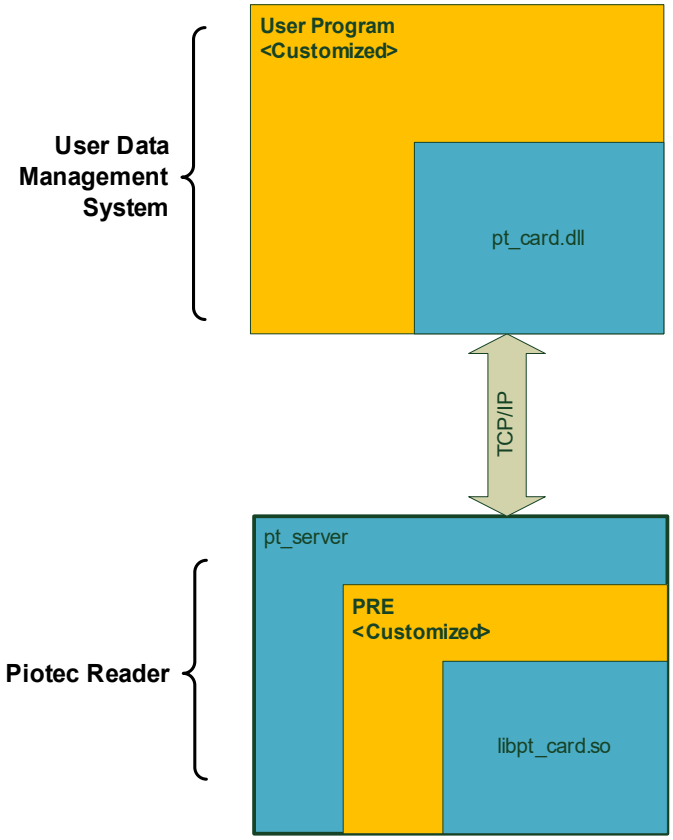

图 2.2 嵌入模式软件结构图

嵌入模式下的各系统组件说明如下：

- pt_card.dll：Piotec 读写器控制模块，通过网络通讯协议与 Piotec 读写器进行指令与数据交互，从而实现对 Piotec 读写器的各类操作。
- pt_server：读写器服务程序，完成与读写器控制模块的网络通讯、读写器内部文件的存储‘PRE’程序的调用。
- PRE：嵌入式写卡应用程序，运行于 Piotec 读写器内部，用于执行写卡指令。在作业初始化时，PRE 程序由数据管理系统 PT_PDM 通过 TCP/IP 网络协议下载到读写器内部并由 pt_server 发起调用。该应用程序允许用户二次开发，且客户可按需指定‘PRE’文件的名称。
- libpt_card.so：读写器驱动程序，运行于 Piotec 读写器内部，为嵌入式写卡应用程序‘PRE’提供读写器操作接口。

第3章 软件工作流程

3.1 指令模式调试

读写器的工作流程包含三部分：初始化、运行和退出。

- 初始化流程：用户程序通过调用 pt_card.dll 的接口函数完成读写器的连接，如图 3.1。

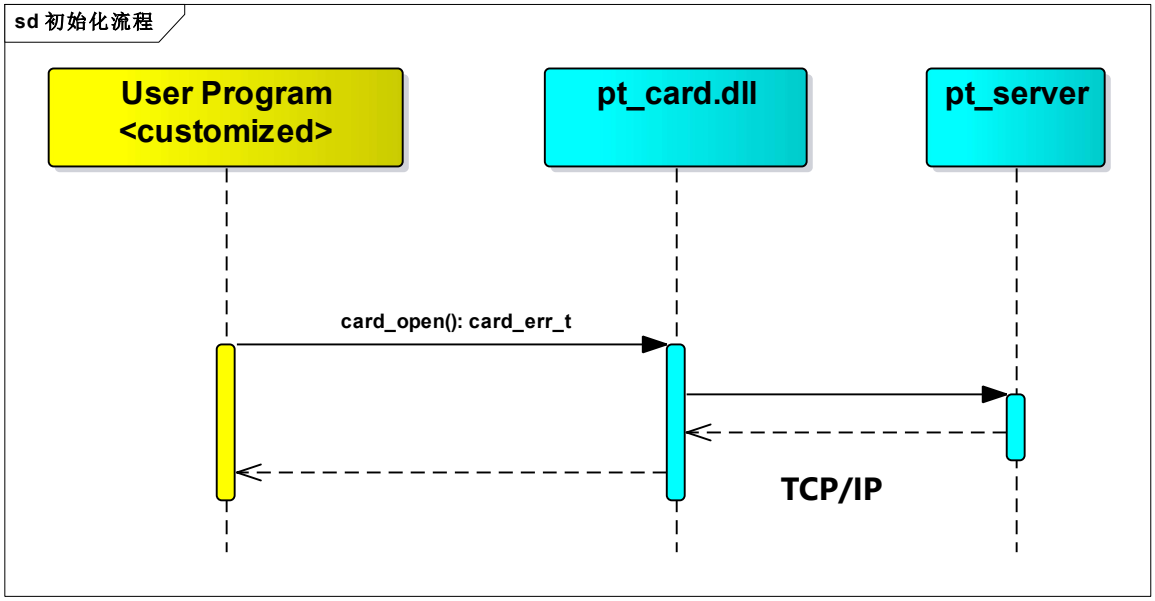

图 3.1 初始化流程

- 运行流程：用户程序通过调用 pt_card.dll 的接口函数完成对芯片的个性化写入，如图 3.2 所示。

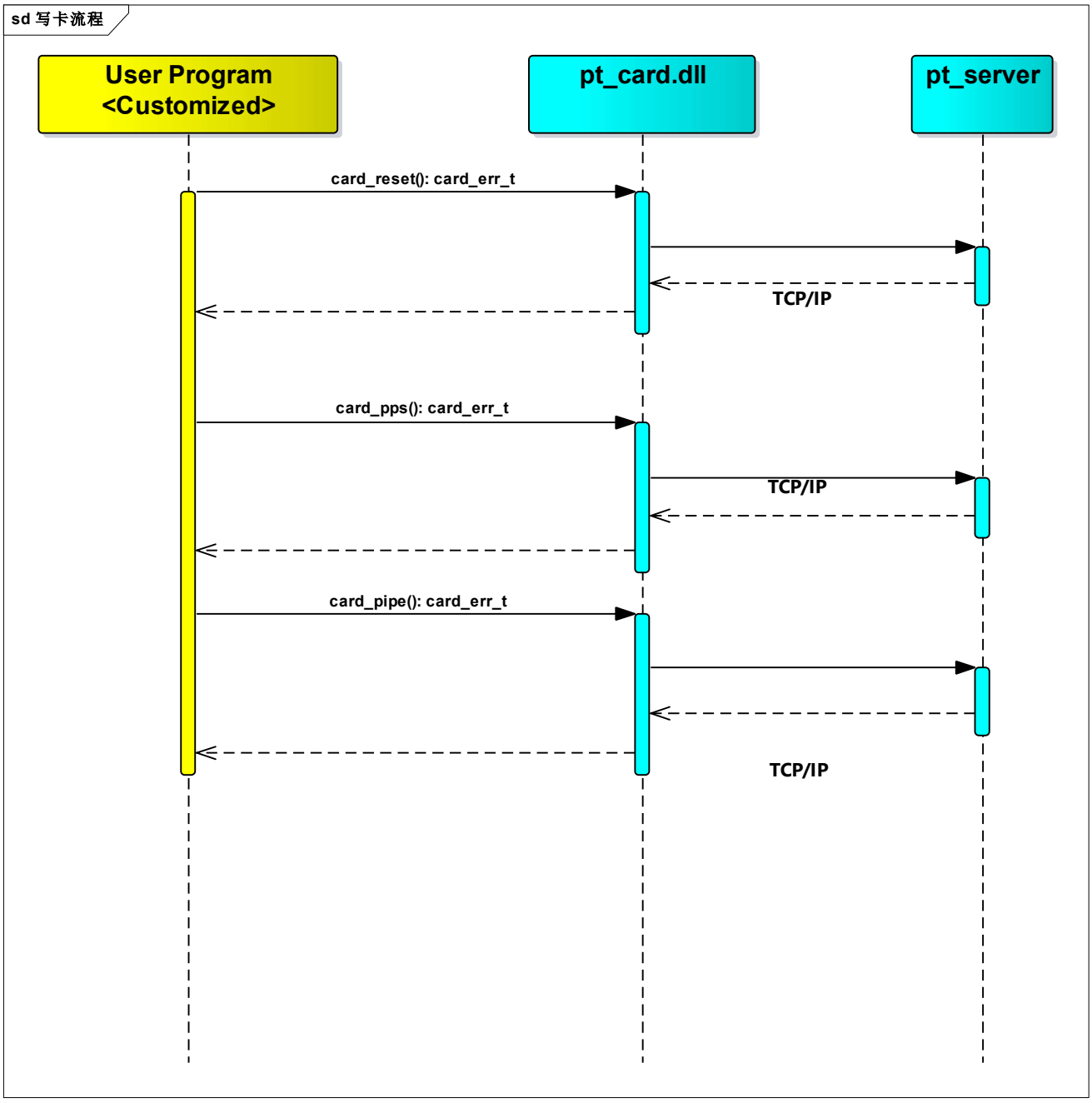

图 3.2 写卡流程

- 退出流程：用户程序通过调用 pt_card.dll 的接口函数完成断开读写器连接等操作，如图 3.3。

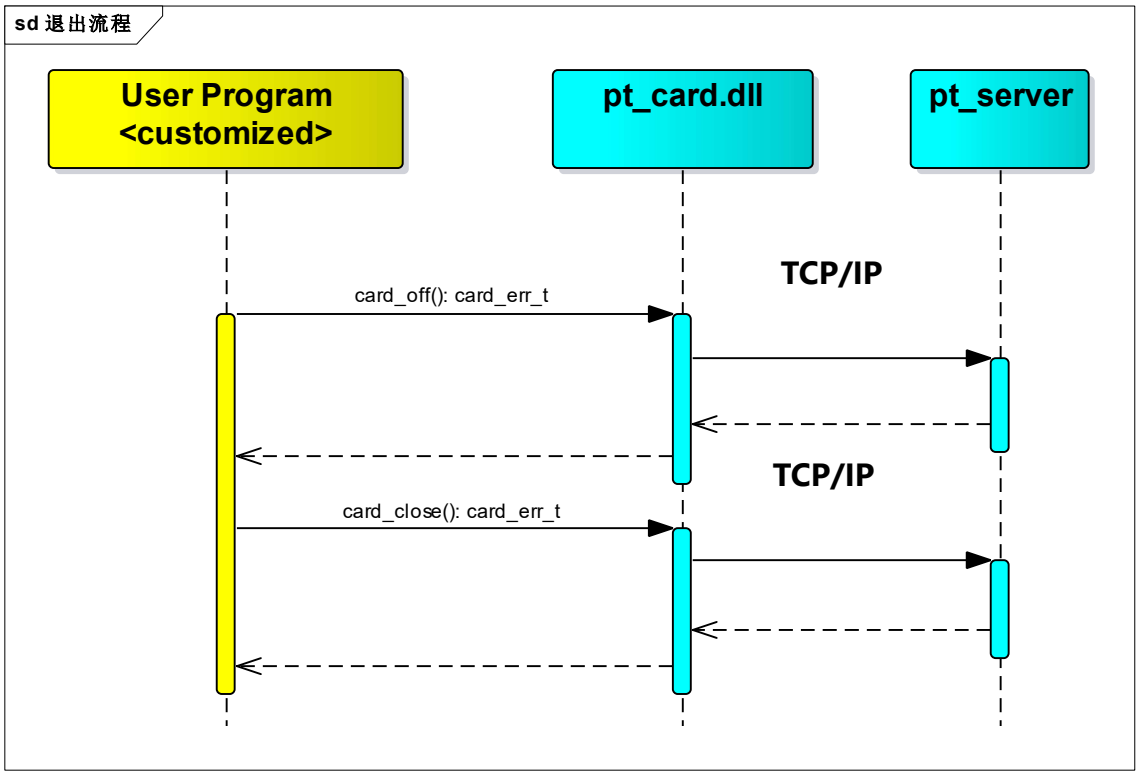

图 3.3 退出流程

3.2 嵌入模式调试

读写器的工作流程包含三部分：初始化、运行和退出。

- 初始化流程：用户程序通过调用 pt_card.dll 的接口函数完成读写器的连接，如图 3.4。

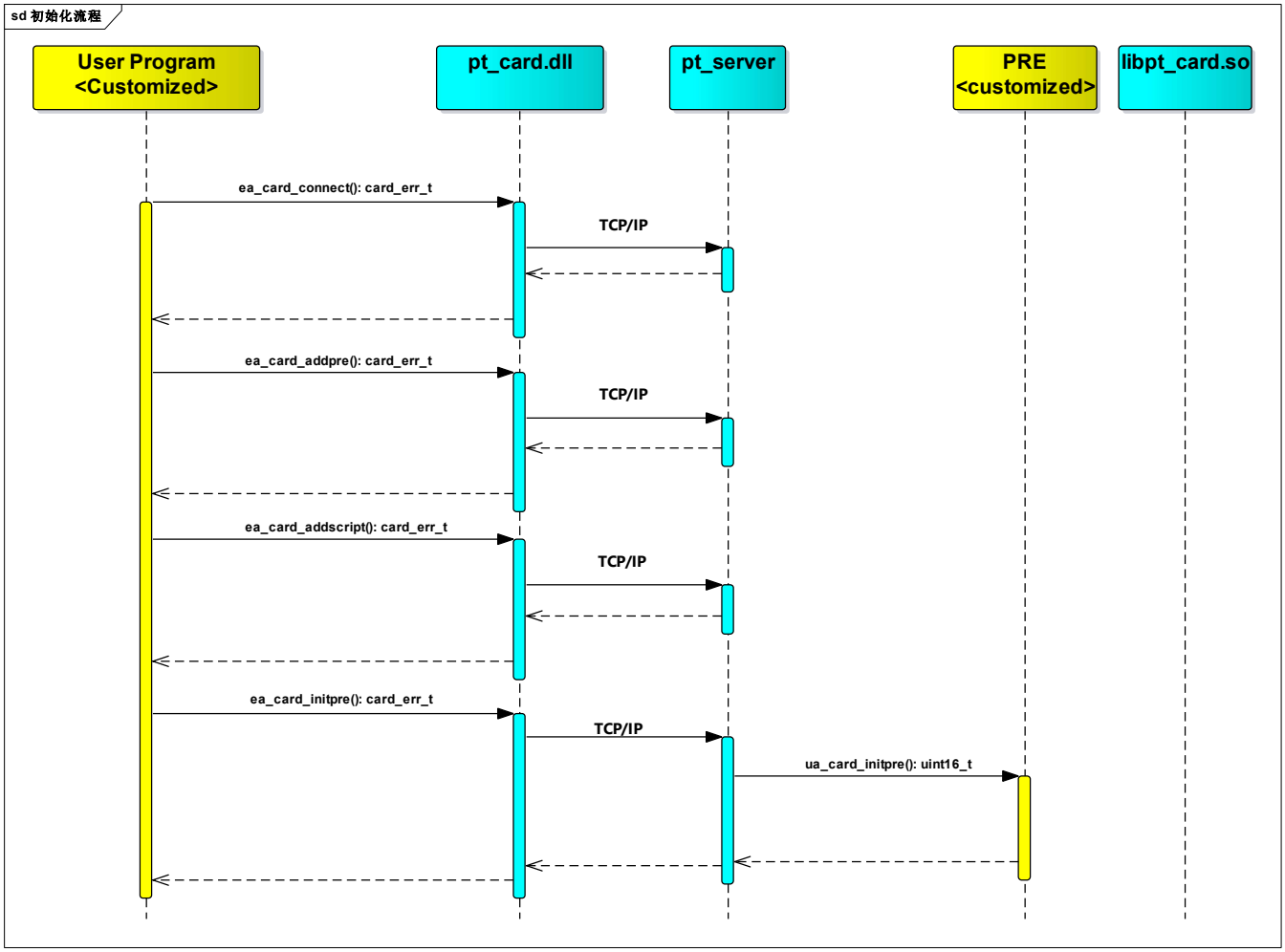

图 3.4 初始化流程

- 运行流程：用户程序通过调用 pt_card.dll 的接口函数完成对芯片的个性化写入，如图 3.5 所示。

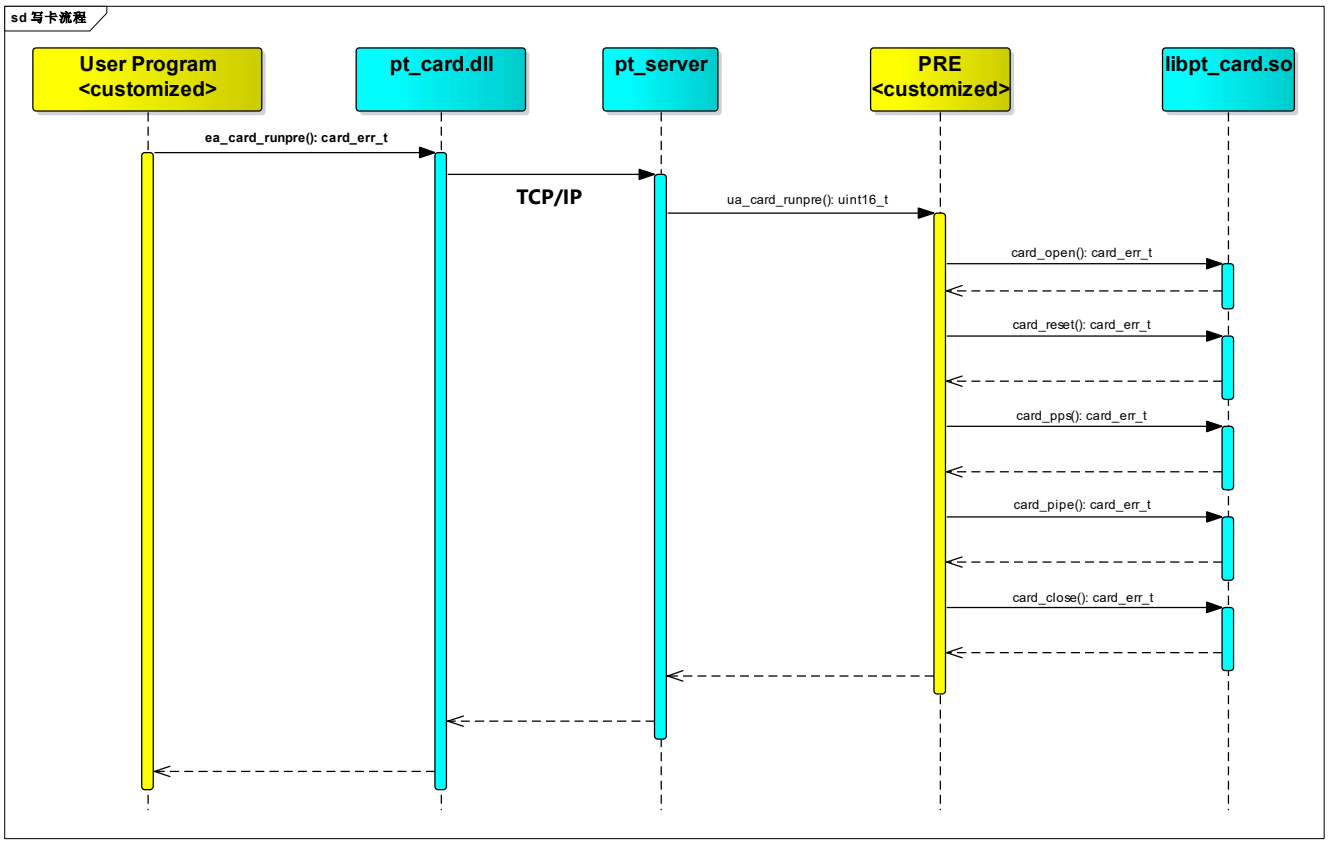

图 3.5 写卡流程

- 退出流程：用户程序通过调用 pt_card.dll 的接口函数完成断开读写器连接等操作，如图 3.6。

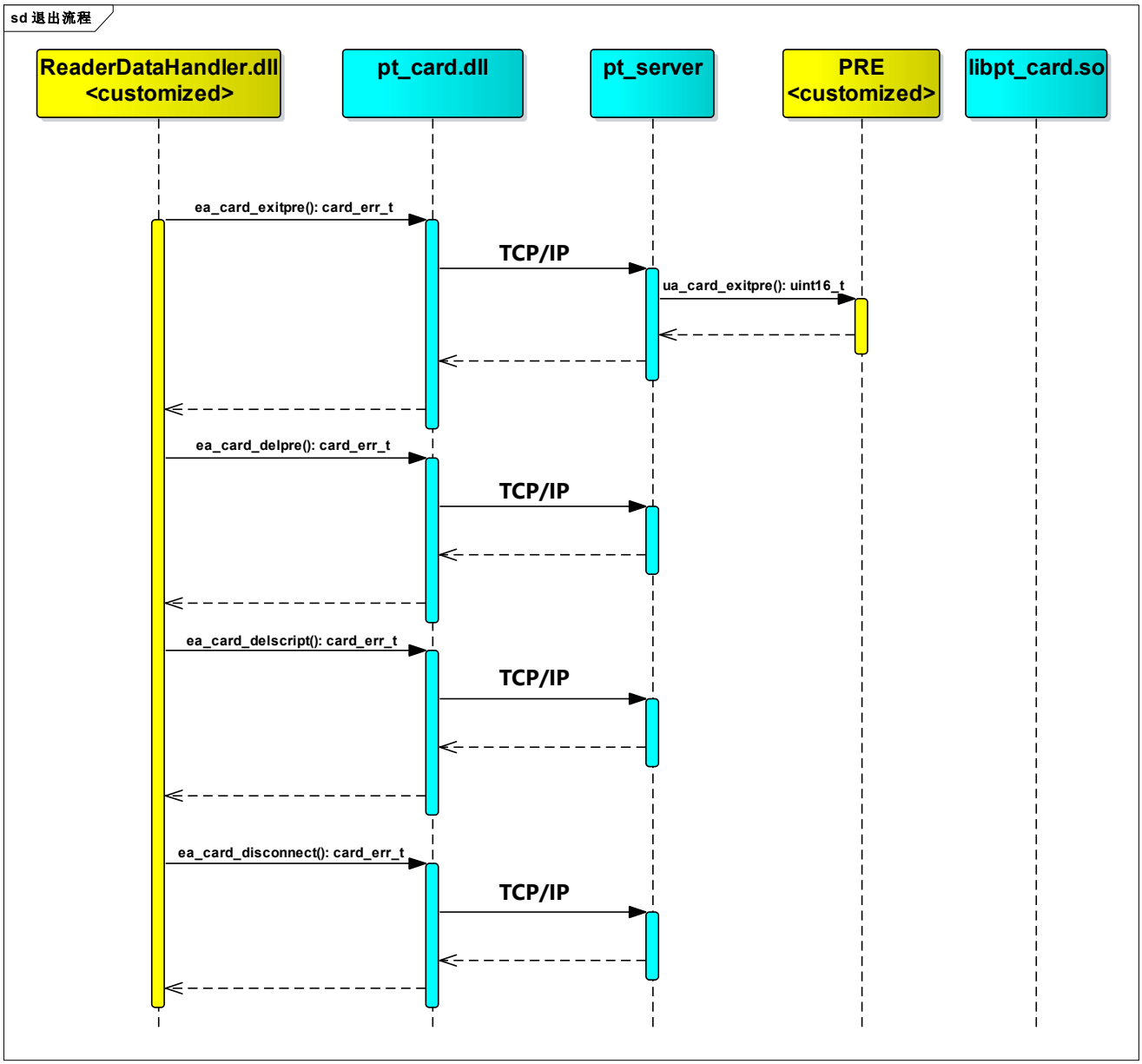

图 3.6 退出流程

第4章 软件接口

Piotec 提供 pt_card.dll 驱动接口用于满足用户的二次开发，用户程序利用 pt_card.dll 提供的接口实现控制指令的下发和智能卡个性化数据写入。

4.1 指令模式通用接口

4.1.1 卡片对象结构体

```
typedefstructcard_object {
    Int32_t handle;                    /*  卡片对象句柄*/
    Uint8_t addr[MAX_ADDR_SIZE];       /*  卡片对象地址*/
    card_mod_tmodel;                   /*  卡片模式*/
    Uint8_t atr[ATR_DATA_LEN];         /*  卡片 ATR 值*/
    Uint8_t atr_len;                   /*  卡片 ATR 长度*/
    Uint8_t sw1;                       /*  卡片返回值 SW1 */
    Uint8_t sw2;                       /*  卡片返回值 SW2 */
    Uint32_t timeout;                  /*  通信超时时限*/
    card_err_tlast_err;                /*  最后出错错误编号*/
} card_obj_t;
```

4.1.2 非接触卡片配置结构体

```
typedefstructcard_picc_config {
    Uint32_t mask;              /** 标记掩码 */
    Uint8_t fsdi;
    Uint8_t fwi;
    Uint32_t fwt;
    Uint8_t dri;
    Uint8_t dsi;
    Uint8_t ds;
    Uint8_t dr;
    Uint16_t sof;
    Uint8_t eof;
```

```
    Uint8_t egt;
    Uint32_t frame_flag;
    Uint16_t system_code;
    Uint8_t nad;
    Uint8_t cid;
    Uint8_t fsci;
    Uint16_t rfon_wait;
    Uint16_t rfoff_wait;
    Uint32_t afdt;
    Uint8_t err_reemit;
    Uint8_t sfgi;
    Uint32_t sfgt;
    Uint8_t afi;
    Uint8_t slot_no;
    Uint16_t tr0tr1;
    Uint32_t afwt;
    Uint32_t asfgt;
} card_pcfg_t;
```

该结构暂时未开放使用。

4.1.3 card_open

- card_err_t card_open（card_obj_t *obj, card_mod_t model, Uint8_t *addr）

功能：初始化卡片通信对象和通信模式。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***model[in]***：卡片模式。见表4.1
3) ***addr[in]***：指令模式：此参数需填入对应读写器IP。

嵌入模式：在使用在线调试模式时，此参数需填入对应读写器IP，正式生产时设置为

NULL即可。

返回值：错误码（如果正确，返回0）。

表 4.1 卡片模式说明表

卡片模式	card_mode_t 枚举定义
ISO14443A	MODEL_P14443A
ISO14443B	MODEL_P14443B
MIFARE	MODEL_PMIFARE
FELICA	MODEL_PFELICA
ISO15693	MODEL_P15693

4.1.4 card_close

- card_err_t card_close（card_obj_t *obj）

功能：卡片下电并关闭读写器，释放资源。

参数：

1) ***obj [in/out]***：卡片对象结构体。

返回值：错误码（如果正确，返回0）。

4.1.5 card_reset

- card_err_t card_reset（card_obj_t *obj）

功能：开启射频载波并执行卡片复位。

参数：

1) ***obj [in/out]***：卡片对象结构体。

返回值：错误码（如果正确，返回0）。

4.1.6 card_on

- card_err_t card_on（card_obj_t *obj）

功能：打开读写器射频载波。

参数：

1) ***obj [in/out]***：卡片对象结构体。

返回值：错误码（如果正确，返回0）。

4.1.7 card_off

- card_err_t card_off（card_obj_t *obj）

功能：关闭读写器射频载波。

参数：

1) ***obj [in/out]***：卡片对象结构体。

返回值：错误码（如果正确，返回0）。

4.1.8 card_pcfg

- card_err_t card_pcfg(card_obj_t *obj, card_pcfg_t *cfg)

功能：非接读写器配置设置。暂时未开放使用

参数：

1) ***obj [in/out]***：卡片对象结构体。

2) ***pcfg [in]***：非接读写器配置结构体。

返回值：错误码（如果正确，返回0）。

4.1.9 card_pps

- card_err_t card_pps（card_obj_t *obj, Uint8_t param1, Uint8_t param2）

功能：设置卡片通信速率。

参数：

1) ***obj [in/out]***：卡片对象结构体。

2) ***param1[in]***：DSI数值。

3) ***param2[in]***：DRI数值。

返回值：错误码（如果正确，返回0）。

表 4.2 卡片通信速率说明表

DSI / DRI	宏定义
106kBit/s	CARD_14443_DATARATE_106

212 kBit/s	CARD_14443_DATARATE_212
424 kBit/s	CARD_14443_DATARATE_424
848 kBit/s	CARD_14443_DATARATE_848

4.1.10 card_pipe

- card_err_t card_pipe（card_obj_t *obj, Uint8_t *tbuf, Uint16_t tlen, Uint8_t *rbuf, Uint16_t *rlen）

功能：读写器与卡片进行数据交换。

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***tbuf [in]***：发送数据缓存。
3) ***tlen [in]***：发送数据长度。
4) ***rbuf [out]***：接收数据缓存。
5) ***rlen [out]***：接收数据长度。

返回值：错误码（如果正确，返回0）。

4.1.11 card_getinfo

- card_err_t card_getinfo（card_obj_t *obj, Uint8_t *info）

功能：获取读写器设备信息。

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***info[out]***：设备信息，字符长度64字节，目前设备信息为版本号。

返回值：错误码（如果正确，返回0）。

4.1.12 card_setfreq

- card_err_t card_setfreq（card_obj_t *obj, Uint16_t freq）

功能：设置读写器写卡频率。**暂时未开放使用**

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***freq[in]***：卡片写卡频率，范围1000~20000，单位KHz。。

返回值：错误码（如果正确，返回0）。

4.1.13 card_getfreq

- card_err_t card_getfreq（card_obj_t *obj, Uint16_t *freq）;

功能：获取读写器写卡频率数值。暂时未开放使用

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***freq[out]***：获取卡片写卡频率数值，单位KHz，范围1000~20000KHz，默认3570KHz。

返回值：错误码（如果正确，返回0）。

4.1.14 card_setmodel

- card_err_t card_setmodel（card_obj_t *obj, card_mod_t model）

功能：设置卡片模式。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***model[in]***：卡片模式。见表4.1

返回值：错误码（如果正确，返回0）。

4.1.15 card_getmodel

- card_err_t card_getmodel（card_obj_t *obj, card_mod_t*model）

功能：获取当前卡片模式。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***model[out]***：卡片模式。见表4.1

返回值：错误码（如果正确，返回0）。

4.1.16 card_automodel

- card_err_t card_automodel（card_obj_t *obj）

功能：自动设置卡片模式。仅支持14443A、14443B和MIFARE卡片。

参数：

1) ***obj[in/out]***：卡片对象结构体。

返回值：错误码（如果正确，返回0）。

4.2 指令模式 ISO14443A 命令接口

4.2.1 card_reqa

- card_err_t card_reqa(card_obj_t *obj, Uint16_t *atqa)

功能：A型卡发送请求命令，在card_on载波开启后使用。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***atqa[out]***：A型卡请求应答，固定2字节长度。

返回值：错误码（如果正确，返回0）。

4.2.2 card_wupa

- card_err_t card_wupa(card_obj_t *obj, Uint16_t *atqa)

功能：A型卡发送唤醒命令，用于调用card_halta停止状态后使用。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***atqa[out]***：A型卡请求应答，固定2字节长度。

返回值：错误码（如果正确，返回0）。

4.2.3 card_halta

- card_err_t card_halta(card_obj_t *obj)

功能：执行A型卡片停止命令，只能通过card_wupa唤醒卡片。

参数：

1) ***obj[in/out]***：卡片对象结构体。

返回值：错误码（如果正确，返回0）。

4.2.4 card_select

- card_err_t card_select(card_obj_t *obj, Uint8_t *uid, Uint16_t uid_len, Uint8_t *sak)

功能：执行卡片选择命令，选择一个已知UID的卡片。

参数：

1) ***obj[in/out]***：卡片对象结构体。

2) ***uid[in]***：卡片唯一标识符。

3) ***uid_len[in]***：卡片唯一标识符长度。

4) ***sak[out]***：选择确认字符。

返回值：错误码（如果正确，返回0）。

4.2.5 card_anticol

- card_err_t card_anticol(card_obj_t *obj, Uint8_t *uid, Uint16_t *uid_len, Uint8_t *sak, Uint8_t *status)

功能：执行防冲突命令并选择卡片，防冲突命令必须在card_reqa或card_wupa后使用。

参数：

1) ***obj[in/out]***：卡片对象结构体。

2) ***uid[in]***：卡片唯一标识符。

3) ***uid_len[in]***：卡片唯一标识符长度。

4) ***sak[out]***：选择确认字符。

5) ***status[out]***：状态标志，0：未检测到冲突、1：检测到冲突。

返回值：错误码（如果正确，返回0）。

4.2.6 card_rats

- card_err_t card_rats(card_obj_t *obj, Uint8_t *ats, Uint16_t *ats_len)

功能：执行卡片请求应答选择命令，激活ISO14443-4层协议。

参数：

1) ***obj[in/out]***：卡片对象结构体。

2) ***ats[out]***：卡片请求应答选择。

3) ***ats_len[out]***：卡片请求应答选择长度。

返回值：错误码（如果正确，返回0）。

4.3 指令模式 ISO14443B 命令接口

4.3.1 card_reqb

- card_err_t card_reqb(card_obj_t *obj, Uint8_t *atqb, Uint8_t *atqb_len)

功能：B型卡发送请求命令，在card_on载波开启后使用。

参数：

1) ***obj[in/out]***：卡片对象结构体。

2) ***atqb[out]***：B型卡请求应答。

3) ***atqb_len[out]***：B型卡请求应答长度。

返回值：错误码（如果正确，返回0）。

4.3.2 card_wupb

- card_err_t card_wupb(card_obj_t *obj, Uint8_t *atqb, Uint8_t *atqb_len)

功能：B型卡发送唤醒命令，用于调用card_haltb停止状态后使用。

参数：

1) ***obj[in/out]***：卡片对象结构体。

2) ***atqb[out]***：B型卡请求应答。

3) ***atqb_len[out]***：B型卡请求应答长度。

返回值：错误码（如果正确，返回0）。

4.3.3 card_haltb

- card_err_t card_haltb(card_obj_t *obj)

功能：执行B型卡片停止命令，只能通过card_wupb唤醒卡片。

参数：

1) ***obj[in/out]***：卡片对象结构体。

返回值：错误码（如果正确，返回0）。

4.3.4 card_attrib

- card_err_t card_attrib(card_obj_t *obj, Uint8_t *rbuf, Uint16_t *rlen)

功能：执行B型卡片attrib命令。

参数：

1) ***obj[in/out]***：卡片对象结构体。

2) ***rbuf[out]***：attrib命令应答数据。

3) ***rlen[out]***：attrib命令应答数据长度。

返回值：错误码（如果正确，返回0）。

4.3.5 card_setattribinf

- card_err_t card_setattribinf(card_obj_t *obj, Uint8_t *tbuf, Uint16_t tlen)

功能：设置B型卡片attrib命令高层配置。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***tbuf[in]***：attrib命令高层配置信息。
3) ***tlen[in]***：attrib命令高层配置信息长度。

返回值：错误码（如果正确，返回0）。

4.3.6 card_getattribinf

- card_err_t card_getattribinf(card_obj_t *obj, Uint8_t *rbuf, Uint16_t *rlen)

功能：获取B型卡片attrib命令高层配置。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***rbuf[out]***：attrib命令高层配置信息。
3) ***rlen[out]***：attrib命令高层配置信息长度。

返回值：错误码（如果正确，返回0）。

4.4 指令模式 ISO14443 通用命令接口

4.4.1 card_getid

- card_err_t card_getid(card_obj_t *obj, Uint8_t *id, Uint8_t *id_len)

功能：获取卡片标识符。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***id[out]***：卡片标识符。ISO14443A：UID、ISO14443B：PUPI。
3) ***id_len[out]***：卡片标识符长度。

返回值：错误码（如果正确，返回0）。

4.4.2 card_deselect

- card_err_t card_deselect(card_obj_t *obj)

功能：发送一个S命令来停用卡。停用14443-4层协议。读写器处于14443-3层协议。执行成功，卡应该在停止状态，只能用card_wupa唤醒激活。

参数：

1) ***obj[in/out]***：卡片对象结构体。

返回值：错误码（如果正确，返回0）。

4.4.3 card_getdetect

- card_err_t card_getdetect(card_obj_t *obj, Uint8_t *val)

功能：检查天线内卡片的存在,读写器R(NAK)块发送,卡片R(ACK)块应答，需要卡片和读写器处于14443-4层协议。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***val[out]***：1字节，1：卡片存在、0：卡片不存在。

返回值：错误码（如果正确，返回0）。

4.5 指令模式 MIFARE 接口

4.5.1 card_authenticate

- card_err_t card_authenticate(card_obj_t *obj, Uint8_t block_no, Uint8_t key_type, Uint8_t *key, Uint8_t *uid)

功能：MIFARE卡片认证。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***block_no[in]***：块编号。
3) ***key_type[in]***：卡片认证密钥类型。
4) ***key[in]***：卡片密钥。密钥为6字节长度。
5) ***uid[in]***：卡片UID。

返回值：错误码（如果正确，返回0）。

表 4.3 卡片模式说明表

MIFARE卡片秘钥类型	宏定义
秘钥A	CARD_MIFARE_KEYA
秘钥B	CARD_MIFARE_KEYB

4.5.2 card_mifare_read

- card_err_t card_mifare_read(card_obj_t *obj, Uint8_t block_no, Uint8_t *rbuf)

功能：MIFARE卡片认证。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***block_no[in]***：块编号。
3) ***rbuf[out]***：读取块数据，长度为16字节。

返回值：错误码（如果正确，返回0）。

4.5.3 card_mifare_write

- card_err_t card_mifare_write(card_obj_t *obj, Uint8_t block_no, Uint8_t *tbuf)

功能：MIFARE卡片认证。

参数：

1) ***obj[in/out]***：卡片对象结构体。
2) ***block_no[in]***：块编号。
3) ***tbuf[in]***：写入块数据，长度为16字节。

返回值：错误码（如果正确，返回0）。

4.6 嵌入模式接口

4.6.1 ea_card_connect

- card_err_t ea_card_connect(card_obj_t *obj, Uint8_t *addr);

功能：连接读写器。

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***addr[in]***：此参数需填入对应读写器IP。

返回值：错误码（如果正确，返回0）。

4.6.2 ea_card_disconnect

- card_err_t ea_card_disconnect(card_obj_t *obj);

功能：断开读写器连接。

参数：

1) ***obj [in/out]***：卡片对象结构体。

返回值：错误码（如果正确，返回0）。

4.6.3 ea_card_addpre

- card_err_t ea_card_addpre(card_obj_t *obj, Uint8_t *path, Uint8_t *name, Uint8_t *md5, Uint8_t force);;

功能：下载PRE程序到读写器中。

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***path[in]***：上位机存储PRE程序路径及程序名。
3) ***name[in]***：读写器内部PRE程序存储名称。
4) ***md5[out]***：下载成功返回PRE文件MD5值（长度16字节）。
5) ***force[in]***：覆盖写入 0-检查文件是否存在 1-不检测文件是否存在,覆盖写入。

返回值：错误码（如果正确，返回0）。

4.6.4 ea_card_addscript

- card_err_t ea_card_addscript(card_obj_t *obj, Uint8_t *path, Uint8_t *name, Uint8_t *md5, Uint8_t force);

功能：下载脚本文件到读写器中。

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***path[in]***：上位机存储脚本文件路径及文件名。
3) ***name[in]***：读写器内部脚本文件存储名称。
4) ***md5[out]***：下载成功返回脚本文件MD5值（长度16字节）。
5) ***force[in]***：覆盖写入 0-检查文件是否存在 1-不检测文件是否存在,覆盖写入。

返回值：错误码（如果正确，返回0）。

4.6.5 ea_card_delpre

- card_err_t ea_card_delpre(card_obj_t *obj, Uint8_t *name);

功能：删除读写器内存储的 PRE 程序。

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***name[in]***：读写器内部存储的PRE程序名称。

返回值：错误码（如果正确，返回0）。

4.6.6 ea_card_delscript

- card_err_t ea_card_delscript(card_obj_t *obj, Uint8_t *name);

功能：删除读写器内存储的脚本文件。

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***name[in]***：读写器内部存储的脚本文件名称。

返回值：错误码（如果正确，返回0）。

4.6.7 ea_card_listpre

- card_err_t ea_card_listpre(card_obj_t *obj, Uint8_t *list, Uint32_t list_len, Uint32_t *list_all_len);

功能：列出读写器中存储的所有PRE程序。

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***list[in]***：PRE程序名称集合缓存,程序名之间用逗号分隔，如”name1,name2”。
3) ***list_len[in]***：PRE程序名称集合缓存长度。
4) ***list_all_len[out]***：PRE程序名称集合实际长度。

返回值：错误码（如果正确，返回0）。

4.6.8 ea_card_listscript

- card_err_t ea_card_listscript(card_obj_t *obj, Uint8_t *list, Uint32_t list_len, Uint32_t *list_all_len);

功能：列出读写器中存储的所有脚本文件。

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***list[in]***：脚本文件名称集合缓存,文件名之间用逗号分隔，如”script1,script2”。
3) ***list_len[in]***：脚本文件名称集合缓存长度。
4) ***list_all_len[out]***：脚本文件名称集合实际长度。

返回值：错误码（如果正确，返回0）。

4.6.9 ea_card_initpre

- card_err_t ea_card_initpre(card_obj_t *obj, Uint8_t *pre_name, Uint8_t *script_name, Uint8_t *user_data, Uint32_t user_data_len, Uint8_t *output_info, Uint32_t *output_info_len);

功能：初始化PRE程序。

参数：

1) ***obj [in/out]***：卡片对象结构体。
2) ***pre_name[in]***：读写器内部PRE程序名称。
3) ***script_name[in]***：读写器内部脚本文件名称。
4) ***user_data[in]***：用户自定义输入数据，如脚本中的关键字信息等。
5) ***user_data_len[in]***：用户自定义输入数据长度。
6) ***output[out]***：用户自定义输出数据，如错误信息等。
7) ***output_info_len[in/out]***：用户自定义输出数据长度,需输入output开辟缓存长度，输出实际输出数据长度。

返回值：错误码（如果正确，返回0）。

4.6.10 ea_card_runpre

- card_err_t ea_card_runpre(card_obj_t *obj, Uint8_t *user_data, Uint32_t user_data_len, Uint8_t *output_info, Uint32_t *output_info_len);

功能：运行PRE程序一次。

参数：

1) ***obj [in/out]***：卡片对象结构体。

2) ***user_data [in]***：用户自定义输入数据，如个性化写入数据。

3) ***user_data_len [in]***：用户自定义输入数据长度。

4) ***output[out]***：用户自定义输出信息，如错误信息等。

5) ***output_info_len[in/out]***：用户自定义输出数据长度,需输入output开辟缓存长度，输出实际输出数据长度。

返回值：错误码（如果正确，返回0）。

4.6.11 ea_card_exitpre

- card_err_t ea_card_exitpre(card_obj_t *obj, Uint8_t *output_info, Uint32_t *output_info_len);

功能：结束PRE程序。

参数：

1) ***obj [in/out]***：卡片对象结构体。

2) ***output[out]***：用户自定义输出信息，如错误信息等。

3) ***output_info_len[in/out]***：用户自定义输出数据长度,需输入output开辟缓存长度，输出实际输出数据长度。

返回值：错误码（如果正确，返回0）。

4.7 指令模式接口调用流程图

4.7.1 通用接口调用流程图

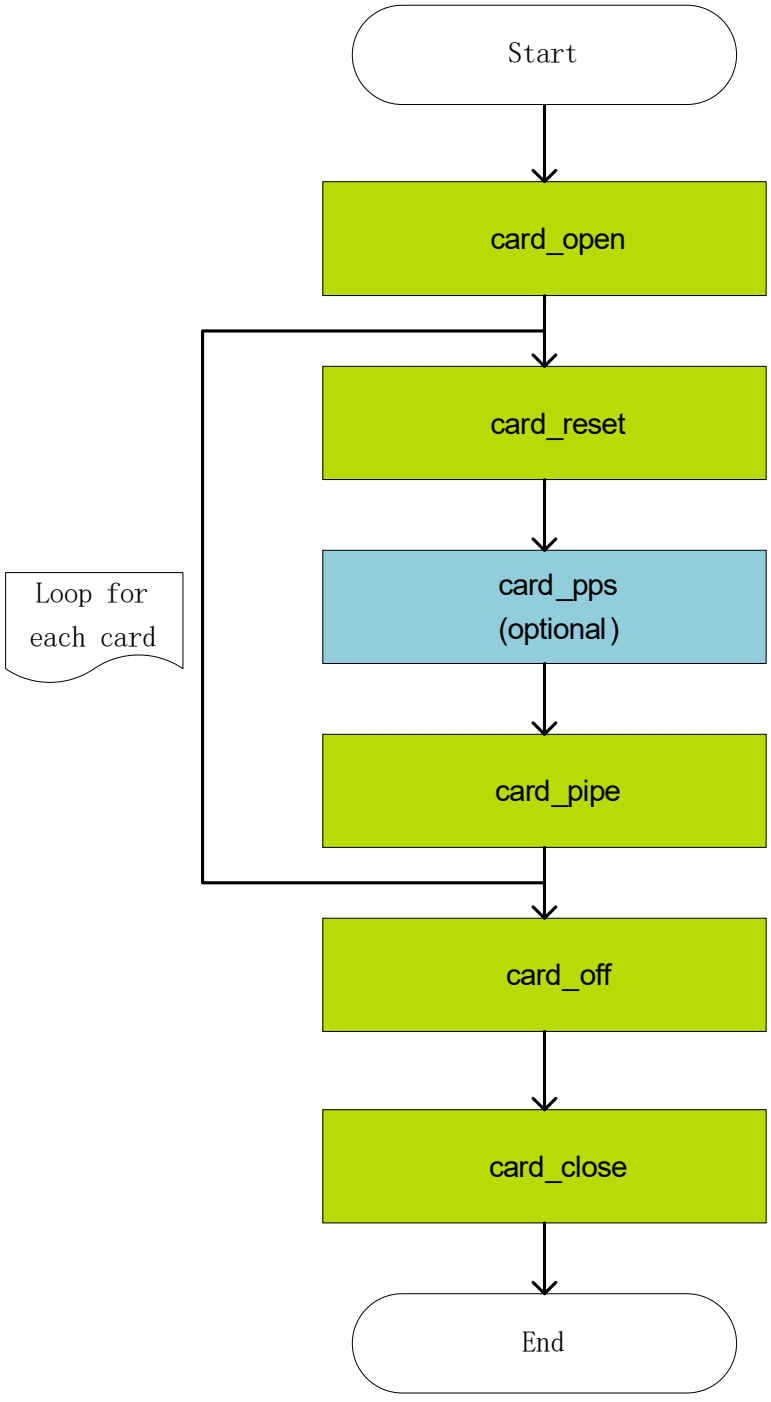

图 4.1 指令模式通用接口调用流程图

4.7.2 ISO14443A 命令接口

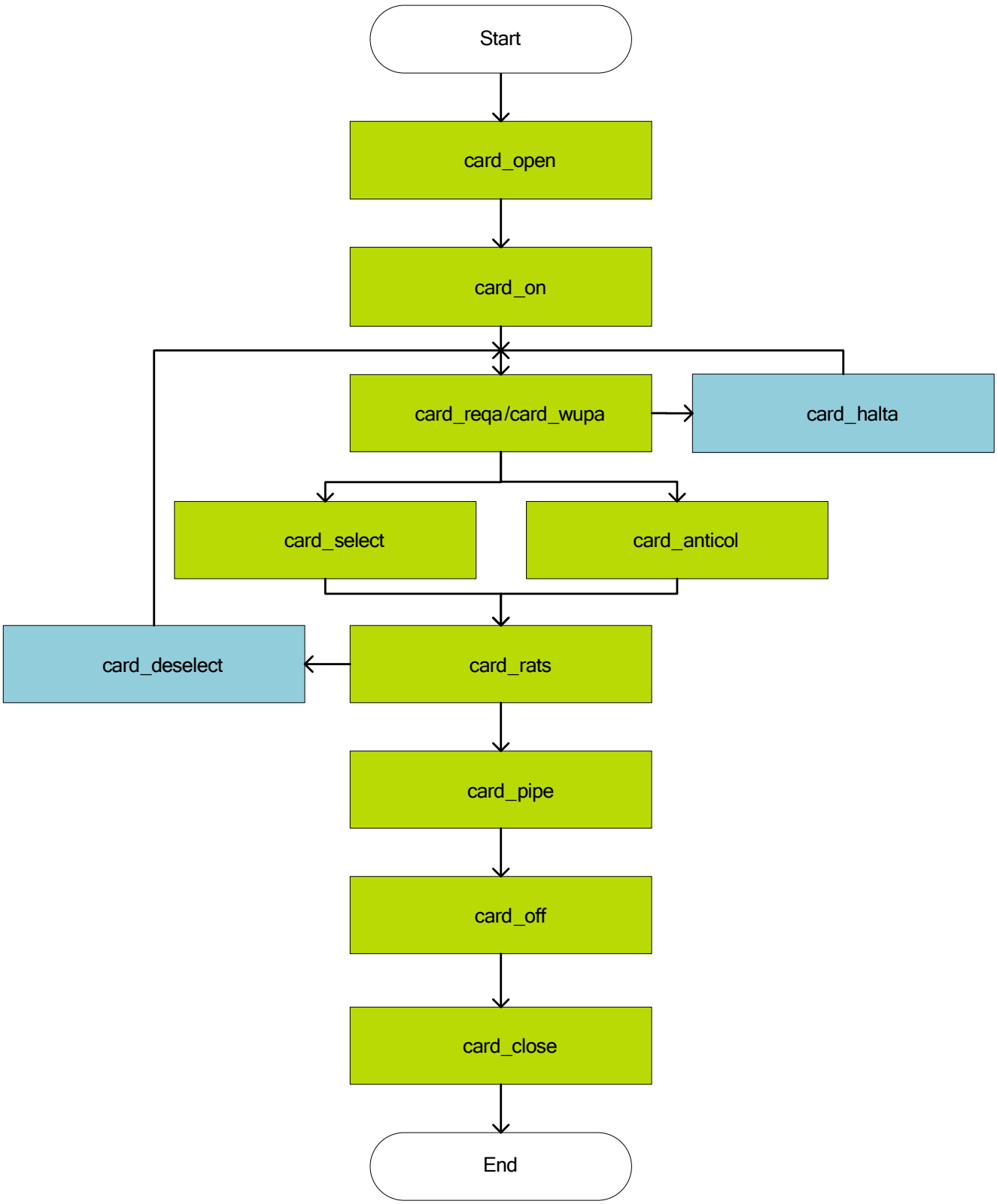

图 4.2 指令模式 14443A 命令接口调用流程图

4.7.3 ISO14443B 命令接口

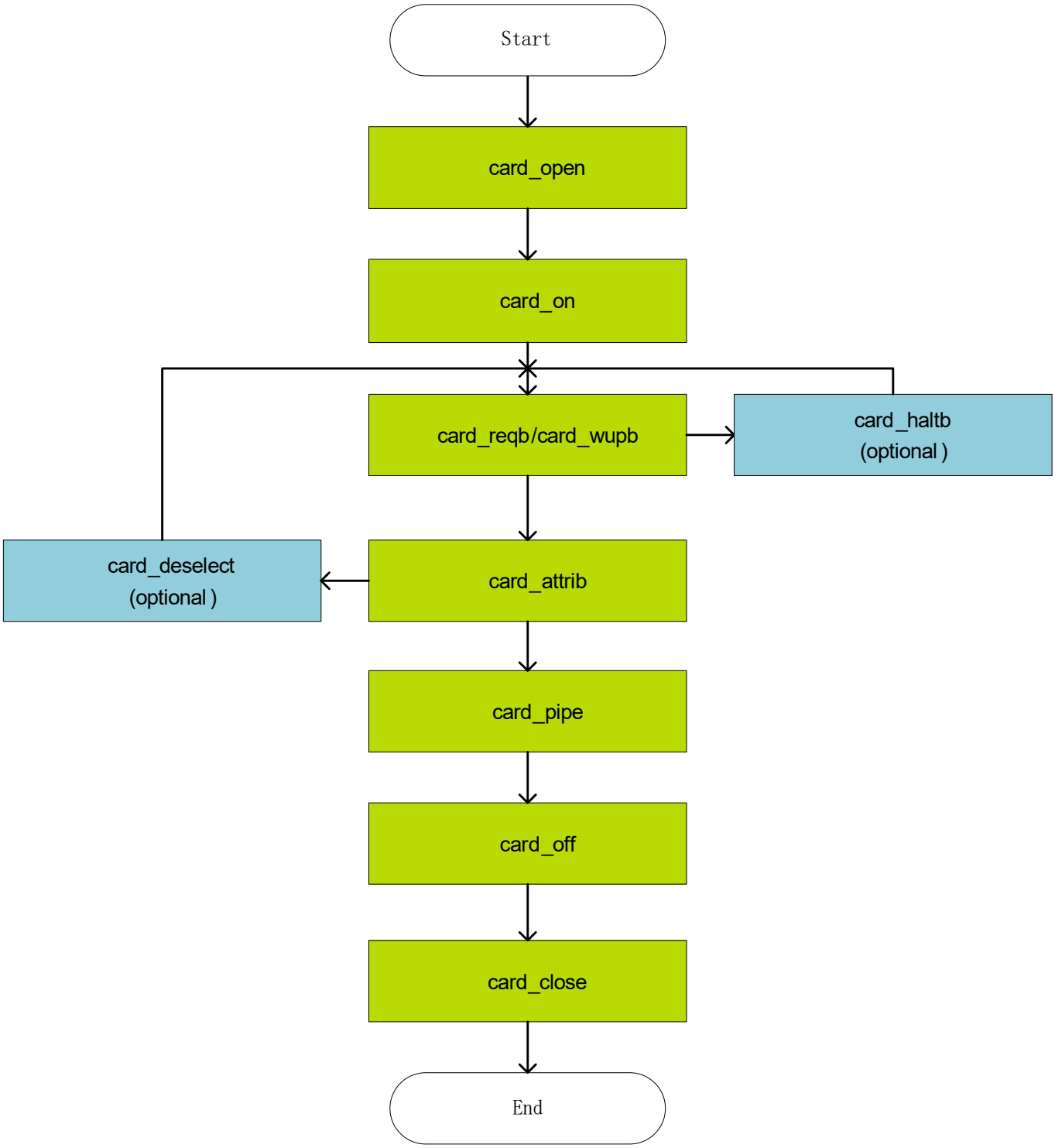

图 4.3 指令模式 14443B 命令接口调用流程图

4.7.4 MIFARE 认证接口

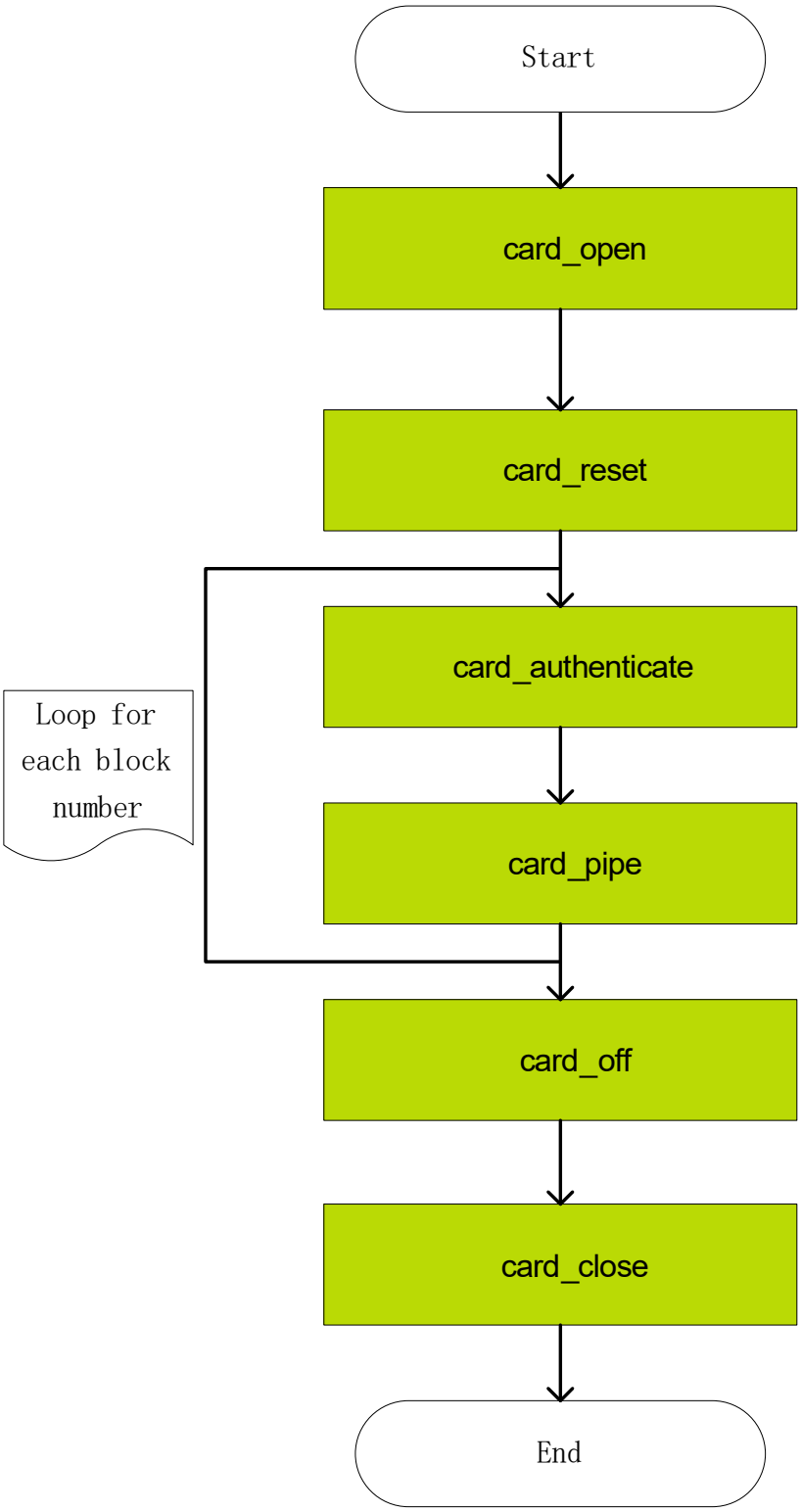

图 4.4 指令模式 MIFARE 认证接口调用流程图

4.8 嵌入模式接口调用流程图

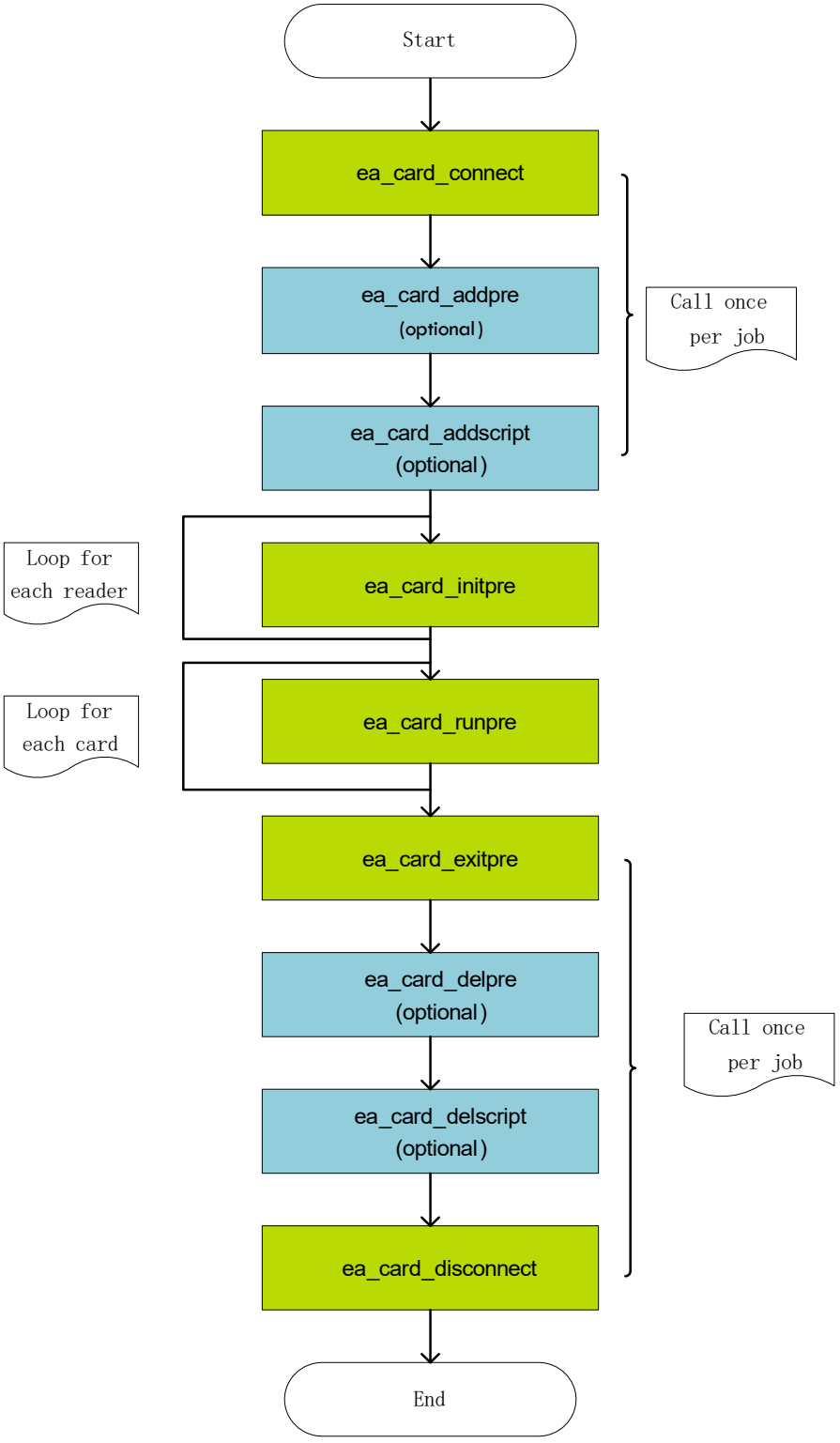

图 4.5 嵌入模式接口调用流程图

4.9 嵌入模式 PRE 程序接口

PRE 程序运行在读写器内部，且该程序需使用固有函数接口模板进行开发，接口函数功能需用户自行开发实现。

<模板程序>

```
#include <iostream>
#include "pt_card.h"
#ifdef __cplusplus
extern "C"{
#endif
using namespace std;
Uint16_t ua_card_initpre（const Uint8_t *script_path, const Uint8_t *user_data, const Uint32_t user_data_len, Uint8_t *output_info, Uint32_t *output_info_len）
{
    return 0;
}
Uint16_t ua_card_runpre（const Uint8_t *user_data, const Uint32_t user_data_len, Uint8_t *output_info, Uint32_t *output_info_len）
{
    return 0;
}
Uint16_t ua_card_exitpre（Uint8_t *output_info, Uint32_t *output_info_len）
{
    return 0;
}
#ifdef __cplusplus
}
#endif
```

4.9.1 ua_card_initpre

- uint16_tua_card_initpre（const Uint8_t *script_path,const Uint8_t *user_data, const Uint32_t user_data_len, Uint8_t *output_info, Uint32_t *output_info_len）

  功能：初始化PRE。

  参数：

  1) ***script_path [in]***：读写器内部存储脚本文件路径及名称。
  2) ***user_data [in]***：用户自定义输入数据，如脚本中的关键字信息等。
  3) ***user_data_len [in]***：用户自定义输入数据长度。
  4) ***output_info [out]***：用户自定义输出信息，如错误信息等。
  5) ***output_info_len [in/out]***：用户自定义输出信息长度,长度不允许超过输入长度。

  返回值：错误码（如果正确，返回0）。

4.9.2 ua_card_runpre

- uint16_t ua_card_runpre（const Uint8_t *user_data, const Uint32_t user_data_len, Uint8_t *output_info, Uint32_t *output_info_len）

  功能：运行PRE。

  参数：

  1) ***user_data [in]***：用户自定义输入数据，如个性化写入数据。
  2) ***user_data_len [in]***：用户自定义输入数据长度。
  3) ***output_info [out]***：用户自定义输出信息，如错误信息等。
  4) ***output_info_len [in/out]***：用户自定义输出信息长度,长度不允许超过输入长度。

  返回值：错误码（如果正确，返回0）。

4.9.3 ua_card_exitpre

- uint16_t ua_card_exitpre（uint8_t *output_info, uint32_t *output_info_len）

  功能：退出 PRE。

  参数：

  1) ***output_info [out]***：用户自定义输出信息，如错误信息等。
  2) ***output_info_len [in/out]***：用户自定义输出信息长度,长度不允许超过输入长度。

返回值：错误码（如果正确，返回0），可用户自定义（0x5000以上）。

4.10 嵌入模式 libpt_card.so 通用接口调用流程图

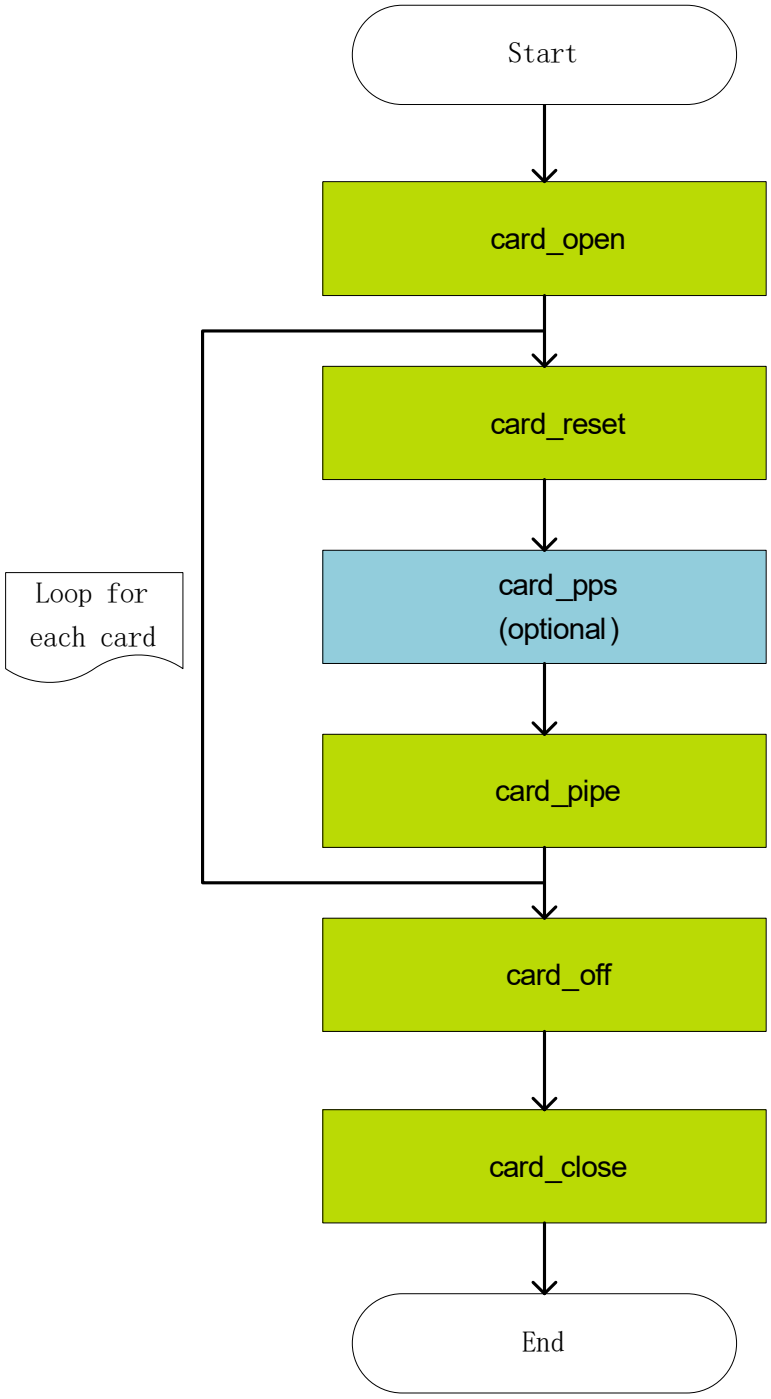

图 4.6 嵌入模式 libpt_card.so 接口调用流程图

第5章 嵌入模式 PRE 程序调试及编译

5.1 PRE 调试说明

由用户二次开发的 PRE 程序，可使用 PIOTEC 提供的“**调试工程 PREDebug**”（存储路径为：\\Piotec reader SDK\ PREDebug）对其进行测试。

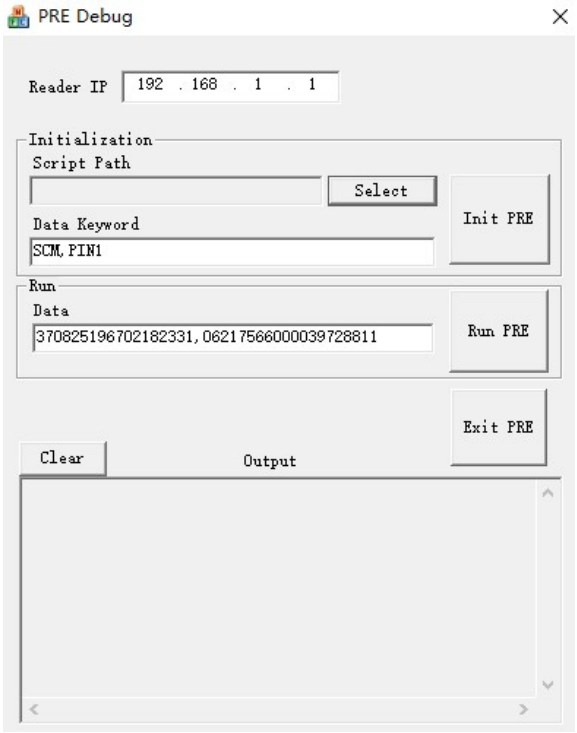

图 5.1 PRE 调试界面

测试过程如下：

1) 使用网线将测试用的 PC 与读写器硬件相连接。
2) 将编写完成的 PRE 程序源代码插入“**调试工程 PREDebug**”中。
3) 运行 PREDebug 程序，界面显示如图 5.1 所示。
4) 在图 5.1 所示界面中的“**Reader IP**”中输入测试读写器的 IP 地址。默认 IP 地址为“192.168.1.1”。
5) Init PRE（ua_card_initpre 接口函数）测试：
   - 单击‘Select’按钮选择需要传入的脚本路径，该路径将被传入 ua_card_initpre 函数的‘*script_path*’参数中。
   - 在‘Data Keyword’文本框中输入数据变量名称，该名称将被传入 ua_card_initpre 函数的‘*user_data*’参数中。
   - 单击‘Init PRE’按钮对 ua_card_initpre 接口函数的调用进行测试。

6) Run PRE（ua_card_runpre 接口函数）测试：

- 在‘Data’文本框中输入个性化数据，该数据将被出传入 ua_card_runpre 函数的‘user_data’参数中。
- 单击‘Run PRE’按钮对 ua_card_runpre 接口函数的调用进行测试。

7) Exit PRE（ua_card_exitpre 接口函数）测试：

- 单击‘Exit PRE’按钮对 ua_card_exitpre 接口函数的调用进行测试。

8) 可通过界面下方的列表框查看测试结果。

5.2 PRE 编译说明

PRE 程序使用 PIOTEC 提供的编译模板工程 PRE Compiler template 进行编译，编译环境为 Linux 平台，需要安装 ARM-LINUX 交叉编译环境。

表 5.1 交叉编译环境

交叉编译环境	说明
安装环境	Ubuntu 12.04.5、Virtualbox
安装包	4.3.3.tgz
编译器版本	gcc version 4.3.3 (Sourcery G++ Lite 2009q1-176)

安装交叉编译环境过程如下：

直接导入方式：

1) 安装 Virtualbox，并导入已配置完成的 Ubuntu 12.04.5 镜像文件 piotec.ova。

重新安装方式：

1) 安装 Virtualbox，并安装 Ubuntu 12.04.5 系统。

2) 登录 Ubuntu 系统，拷贝交叉编辑器安装包 4.3.3.tgz 至/usr/local，并解压安装包至此文件夹。

相关命令：

```
cp toolchain.tar.gz /usr/local
cd /usr/local
tar -xzvf toolchain.tar.gz
```

3) 配置环境变量，使其生效。

相关命令：

vim /etc/bash.bashrc

//在最后添加

PATH=$PATH:/usr/local/4.3.3/bin

//保存退出

输入**:wq**

//使其立即生效

source /etc/bash.bashrc

4) 输入 arm- linux- gcc–v，显示版本号：gcc version 4.3.3 (Sourcery G++ Lite 2009q1-203) ，说明此次安装成功。

编译模板工程 PRE Compiler template（存储路径为：\\Piotec reader SDK\ PRE Compiler template），该工程包含以下文件及目录：

表 5.2 编译模板

目录及文件名称	说明
include 目录	包含调用需要的 pt_card.h 文件
lib 目录	包含调用需要的 libpt_card.so 文件
PRE 目录	存放编译完成的用户 PRE 程序
src 目录	存放用户 PRE 程序源代码 目录包含模板源文件 sample.cpp
Makefile 文件	执行编译文件

编译过程如下：

1) 把调试完成的用户 PRE 程序源代码存放到 src 目录下，例如：src 目录下 sample.cpp。

2) 在工程目录下执行“**make SRC=用户 PRE 源程序文件名**”，例如：make SRC=sample。

3) 若编译成功，PRE 目录下将自动生成用户 PRE 程序，例如：PRE 目录下 sample.pre。

第6章 调试

用户在进行二次开发时，可使用 PIOTEC 提供的“PTCtlsReaderTest 调试工程”对其调用 pt_card.dll 的过程进行测试。

6.1 指令模式调试

6.1.1 基本指令模式

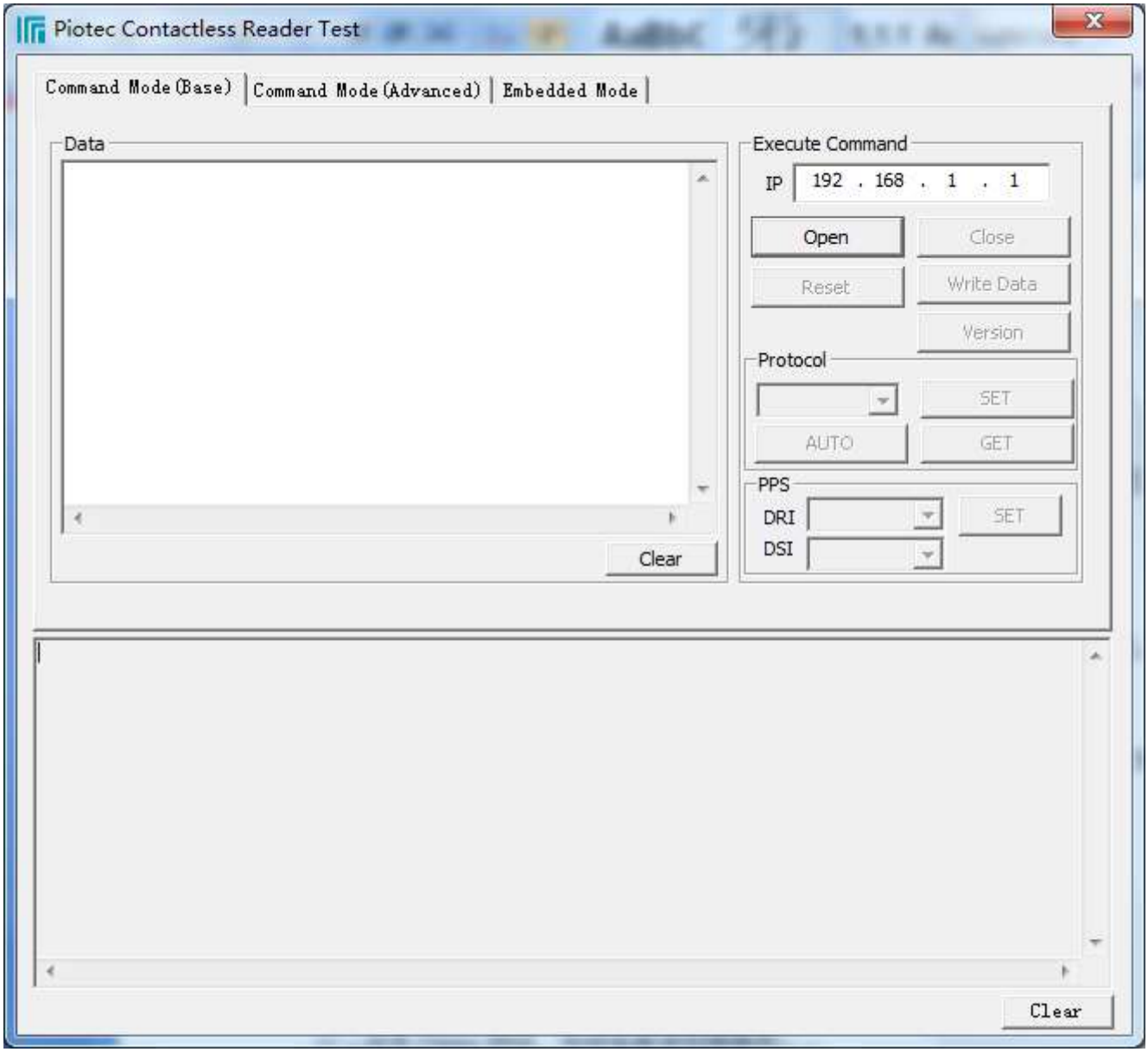

图 6.1 基本指令模式调试

写卡测试过程如下：

1） 在“IP 地址栏”输入读写器 IP 地址。

2） 点击“Open”按钮，完成连接读写器操作。默认使用 14443A 协议。

3） 在“Protocol”复选框内选择需要设定的通信协议，例如“14443A”。点击“Set”按钮，完成对读写器通信协议的设置。（可选）

4） “AUTO”按钮可以自动设置通信协议，目前支持 14443A、14443B 和 MIFARE 卡片。（可选）

5） 点击“Reset”按钮，完成对卡片复位操作。

6） 在“PPS”中的“DSI”和“DRI”复选框选择需要设定的通信速率数值，例如“106 kBit/s”。点击“Set”按钮，完成对读写器通信速率的设置（可选）

7） 在左侧“Data”输入框填入写卡的 APDU 指令，点击“Write Data”按钮，完成对卡片的写卡操作。

8） 点击“Close”按钮，断开读写器。

6.1.2 高级指令模式

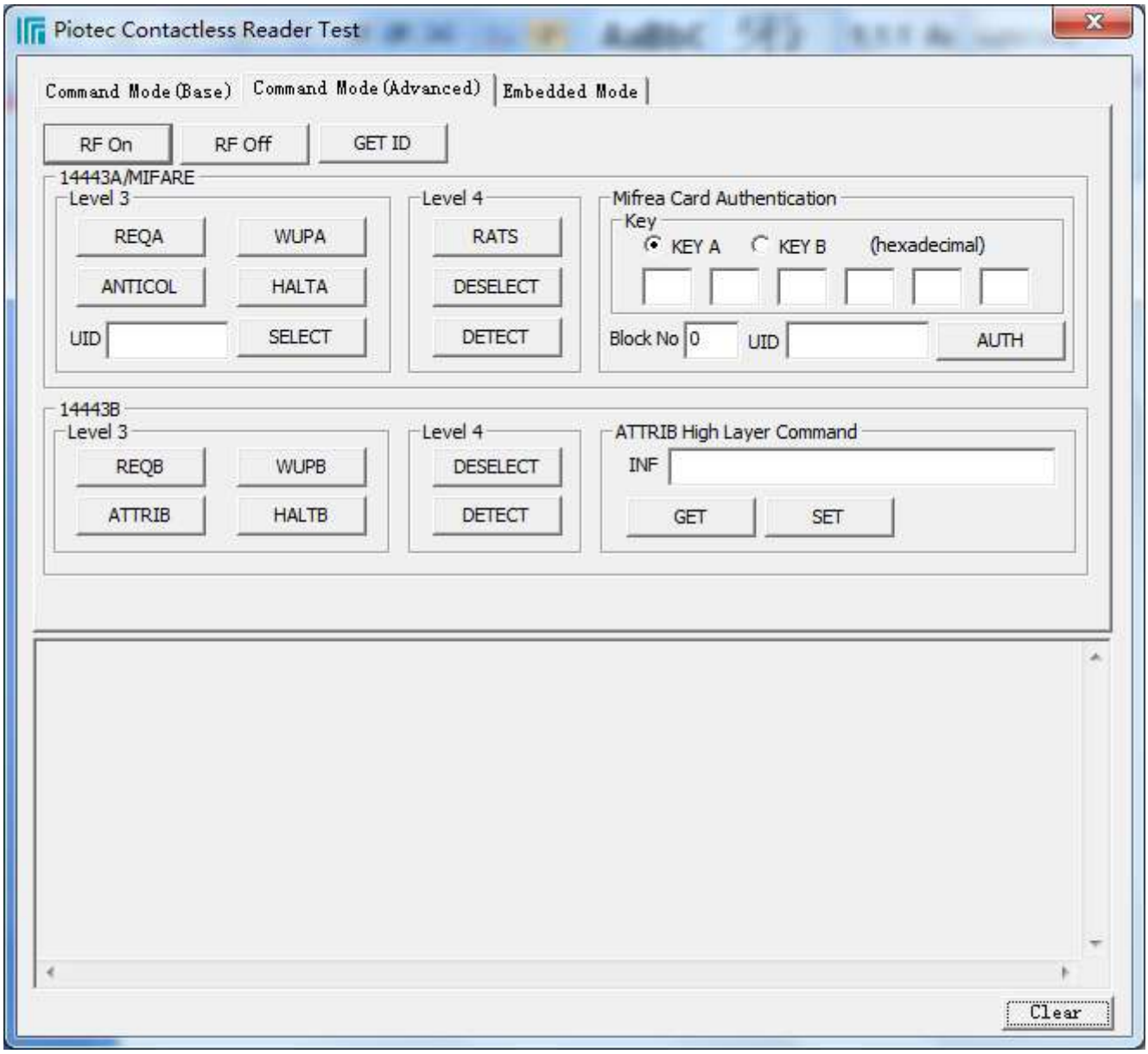

图 6.2 高级指令模式调试

14443A 测试过程如下：

1) 在基本模式界面下，在“IP 地址栏”输入读写器 IP 地址，点击“Open”按钮，完成连接读写器操作。默认使用 14443A 协议。

2) 点击“RF On”按钮，打开射频载波操作。

3) 点击“REQA”按钮，完成对卡片发送 A 类卡的请求操作。

4) 点击“HALTA”按钮，完成对卡片的休眠操作。（可选）

5） 点击“WUPA”按钮，完成对卡片的唤醒操作。（可选）

6） 点击“ANTICOL”按钮，完成对卡片的防碰撞及选择操作。或者在“UID 编辑框”输入卡片 UID 并点击“SELECT”按钮，完成对卡片的选择操作。激活 ISO14443-3 层协议。

7） 点击“RATS”按钮，完成对卡片的请求应答选择命令，激活 ISO14443-4 层协议。

8） 点击“DETECT”按钮，查看卡片是否处于 14443-4 层协议。（可选）

9） 点击“DESELECT”按钮，发送 S 命令操作，停用 14443-4 层协议。（可选）

10）点击“RF Off”按钮，关闭射频载波操作。

11）在基本模式界面下，点击“Close”按钮，断开读写器。

MIFARE 测试过程如下：

1） 在基本模式界面下，在“IP 地址栏”输入读写器 IP 地址，点击“Open”按钮，完成连接读写器操作。在“Protocol”复选框内选择选择 MIFARE 协议并设置。

2） 在基本模式界面下，点击“Reset”按钮，完成对卡片的复位操作。

3） 在“Mifare Card Authentication”中选择秘钥类型（A 类或 B 类），并填入秘钥（16 进制）、块编号和 UID，点击“AUTH”完成认证。如图：

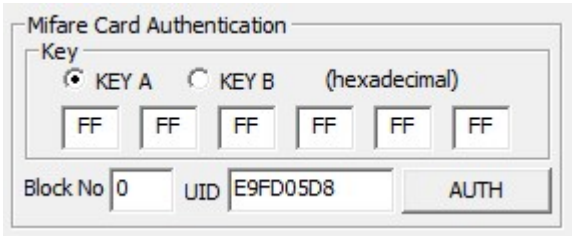

图 6.3 高级指令模式 MIFARE 认证

4） 基本模式界面下，点击“Close”按钮，断开读写器。

14443B 测试过程如下：

1） 基本模式界面下，在“IP 地址栏”输入读写器 IP 地址，点击“Open”按钮，连接读写器。在“Protocol”复选框内选择选择 14443B 协议并设置。

2） 点击“RF On”按钮，打开射频载波操作。

3） 点击“REQB”按钮，完成对卡片的发送 B 类卡请求操作。

4） 点击“HALTB”按钮，完成对卡片的休眠操作。（可选）

5） 点击“WUPB”按钮，完成对卡片的唤醒操作。（可选）

6） 点击“ATTRIB”按钮，执行 B 型卡片 attrib 命令，激活 ISO14443-4 层协议。

7） 点击“DETECT”按钮，查看卡片是否处于 14443-4 层协议。（可选）

8） 点击“DESELECT”按钮，发送 S 命令操作，停用 14443-4 层协议。（可选）

9） 点击“RF Off”按钮，关闭射频载波操作。

10）在基本模式界面下，点击“Close”按钮，断开读写器。

6.2 嵌入模式调试

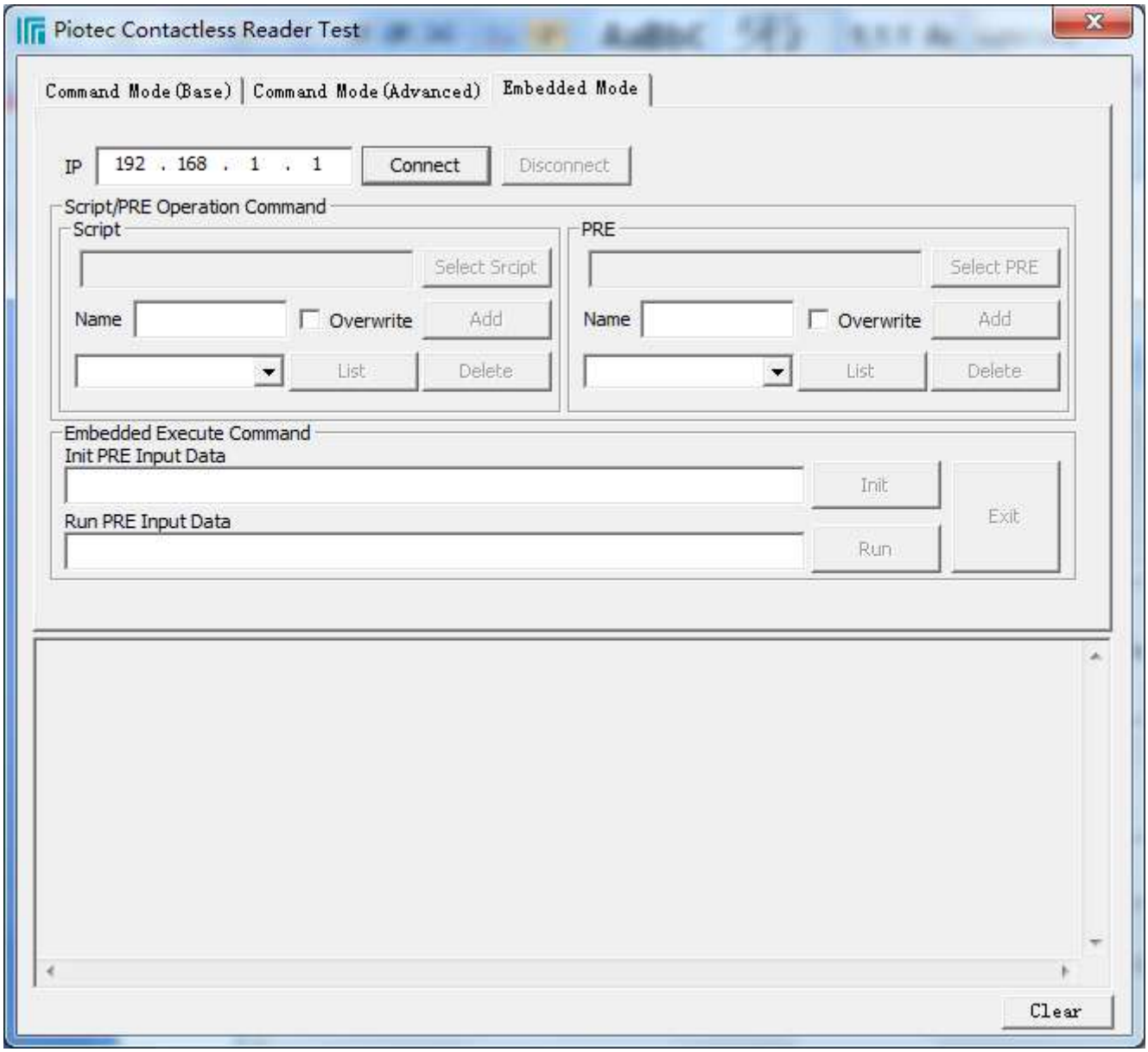

图 6.4 嵌入模式调试

写卡测试过程如下：

1） 在“IP 地址栏”输入读写器 IP 地址。

2） 点击“Connect”按钮，完成连接读写器操作。

3） 点击“Select Script”按钮和“Select PRE”按钮，选择需要下载的脚本文件和 PRE 文件。

4） 在“Name 编辑框”中填入需要下载的脚本文件和 PRE 文件的名称，点击“Add”按钮进行文件

下载操作（Overwrite 选项表示重名文件是否覆盖）。

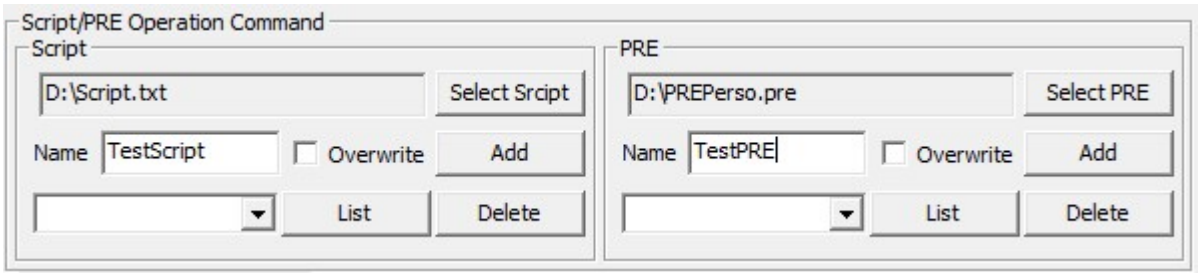

图 6.5 嵌入模式脚本操作界面

5） 点击“List”按钮列出读写器内部保存的脚本文件和 PRE 文件。点击“Delete”按钮删除复选框中对应的文件。

图 6.6 嵌入模式删除脚本

6） 嵌入式初始化过程，在复选框中选择需要使用的脚本文件名和 PRE 文件名，在“Init PRE Input Data”编辑框中填入需要初始化输入的数据（可选），点击“Init”按钮进行嵌入式模式初始化操作。

7） 嵌入式运行过程，在“Run PRE Input Data”编辑框中填入需要运行时输入的数据（可选），点击“Run”按钮进行嵌入式模式运行操作。

8） 嵌入式释放过程，点击“Exit”按钮进行嵌入式模式释放操作。

9） 点击“Disconnect”按钮，断开读写器。

第7章 错误码说明

表 7.1 错误码说明

错误码	说明
0x0000	无错误
0x1001	无效的电压参数
0x1002	上电中断
0x1003	ATR 超时
0x1004	无效的 ATR 值
0x1005	ATR 校验失败
0x1006	协议不支持
0x1007	无效的 SLOT
0x1008	接收数据超时
0x1009	接收数据奇偶校验失败
0x1010	接收 LRC 失败
0x1011	接收 CRC 失败
0x1012	接收序号错误
0x1013	设置 IFSD 失败
0x1014	热复位错误
0x1015	非法的状态字节
0x1016	未知的错误
0x1017	块等待超时

0x1018	字符等待超时
0x1019	T1 协议终止传输错误
0x1020	超过最大重试次数
0x1021	无效的 NAD
0x1022	无效的的长度
0x1023	无效的信息
0x1024	无效的工作模式
0x1025	无效的等级
0x1026	未知的控制命令
0x1027	不支持 PPS 交换
0x1028	未选择 SLOT 模式
0x1029	工作模式错误
0x1030	上电模式错误
0x1031	电压错误
0x1032	卡片类型错误
0x1033	协议错误
0x1034	PPS 模式错误
0x1035	接收数据溢出
0x1036	发送数据超时
0x1037	无效参数

0x1038	设备忙
0x1039	获取用户数据失败
0x1040	返回用户数据失败
0x1041	设置频率失败
0x1042	设置 WWT 失败
0x1043	设置忽略 SW 失败
0x1044	开短路测试失败
0x2001	接收数据无应答超时
0x2002	CRC 或奇偶校验错误
0x2003	发生碰撞
0x2004	缓存溢出
0x2005	无效的帧格式
0x2006	应答违反协议
0x2007	验证失败
0x2008	内存读写错误
0x2009	RC 温度过热
0x2010	RF 错误
0x2011	RC 通信错误
0x2012	长度错误
0x2013	设备资源错误

0x2014	发送端 NAK 错误
0x2015	接收端 NAK 错误
0x2016	外部 RF 错误
0x2017	EMVCoEMD 噪声错误
0x2018	忽略错误
0x2019	外部错误
0x2020	无效的数据参数
0x2021	无效参数
0x2022	读写参数溢出
0x2023	参数不支持
0x2024	命令不支持
0x2025	条件不支持
0x2026	KEY 错误
0x2027	初始化错误
0x3001	设备已打开
0x3002	打开设备失败
0x3003	关闭设备失败
0x3004	IP 错误
0x3005	功能不支持
0x3006	协议不支持

0x3007	参数错误
0x3008	文件已存在
0x3009	打开文件失败
0x3010	读取文件失败
0x3011	写入文件失败
0x3012	检测 MD5 值失败
0x3013	读写器型号错误
0x4001	未知通信错误
0x4002	IP 地址为空
0x4003	通信连接未打开
0x4004	通信连接打开失败
0x4005	通信连接初始化失败
0x4006	通信连接关闭失败
0x4007	通信连接中断
0x4008	通信连接超时
0x4009	发送数据失败
0x4010	接收数据失败
0x4011	数据命令不支持
0x4012	内存错误

0x4013	CRC 校验失败
0x4014	未知应答
0x4015	应答数据长度错误
0x4016	网络断开
0x4017	参数错误
0x4018	通信连接已打开

--- End of document ---

PIOTEC

沈阳派尔泰科科技有限公司

地址：沈阳市浑南新区世纪路37号
电话：024-23783416　传真：024-23783416
邮箱：sales@piotec.cn　网址：www.piotec.cn