

PIOTEC Contactless Reader Drive Interface

Development Manual

沈阳派尔泰科科技有限公司

Document Information

Document name	Development Manual for PIOTEC Contactless Reader Drive Interface				
Category	Product Document				
Document Number	PT-READER-WD004-002				
File Specification					
REVISION HISTORY					
Version	Date	Chapter	Type	Author	Abstract
V1.0	2017/12/14	Whole document	Establishment	Xiaofan Feng	
V1.1	2017/12/16	Whole document	Revision	Qingguo Xiu	
V1.2	2018/01/16		Revision	Xiaofan Feng	Add location description of configuration files
V1.3	2018/01/19		Revision	Xiaofan Feng	Add description of software interface

V1.4	2018/01/23		Revision	Xiaofan Feng	Modify description of MIFARE card authentication function parameters
------	------------	--	----------	-----------------	---

Contents

Chapter 1 Introduction.....	1
1.1 Purpose	2
1.2 Applicable Production	2
1.3 Glossary.....	2
1.4 SDK Directory Structure	2
Chapter 2 Software Structure	4
2.1 Software Structure of Command Mode.....	5
2.2 Software Structure of Embedded Mode	5
Chapter 3 Software Workflow	8
3.1 Debugging of Command Mode	9
3.2 Debugging of Embedded Mode.....	12
Chapter 4 Software Interface	15
4.1 Universal Interface of Command mode.....	16
4.1.1 Card object Structure	16
4.1.2 Contactless Card Configuration Structure	16
4.1.3 card_open	17
4.1.4 card_close	18
4.1.5 card_reset	18
4.1.6 card_on	18
4.1.7 card_off	19
4.1.8 card_pcfg.....	19
4.1.9 card_pps.....	19
4.1.10 card_pipe.....	20
4.1.11 card_getinfo	20
4.1.12 card_setfreq	20
4.1.13 card_getfreq	21

4.1.14	card_setmodel.....	21
4.1.15	card_getmodel	21
4.1.16	card_automodel	22
4.2	ISO14443A Command Interface of Command Mode	22
4.2.1	card_reqa	22
4.2.2	card_wupa.....	22
4.2.3	card_halta.....	22
4.2.4	card_select.....	23
4.2.5	card_anticol	23
4.2.6	card_rats	23
4.3	ISO14443B Command Interface of Command Mode	24
4.3.1	card_reqb	24
4.3.2	card_wupb.....	24
4.3.3	card_haltb.....	25
4.3.4	card_attrib	25
4.3.5	card_setattribinf.....	25
4.3.6	card_getattribinf	25
4.4	ISO14443 Universal Command Interface of Command mode	26
4.4.1	card_getid.....	26
4.4.2	card_deselect.....	26
4.4.3	card_getdetect	26
4.5	MIFARE Authentication Interface of Command Mode	27
4.5.1	Card_ authenticate.....	27
4.6	Embedded Mode Interface	27
4.6.1	ea_card_connect.....	27
4.6.2	ea_card_disconnect.....	28
4.6.3	ea_card_addpre	28

4.6.4	ea_card_addscript.....	28
4.6.5	ea_card_delpre	29
4.6.6	ea_card_delscript.....	29
4.6.7	ea_card_listpre.....	29
4.6.8	ea_card_listscript	29
4.6.9	ea_card_initpre	30
4.6.10	ea_card_runpre.....	30
4.6.11	ea_card_exitpre	31
4.7	Interface Call Flow Chart of Command Mode	32
4.7.1	Universal Interface Call Flow Chart	32
4.7.2	ISO14443A Command Interface	33
4.7.3	ISO14443B Command Interface.....	34
4.7.4	MIFARE Authentication Interface	35
4.8	Interface Call Flow Chart of Embedded Mode	36
4.9	PRE Program Interface of Embedded Mode.....	37
4.9.1	ua_card_initpre	38
4.9.2	ua_card_runpre.....	38
4.9.3	ua_card_exitpre	38
4.10	libpt_card.so Universal Interface Call Flow Chart of Embedded Mode	39
Chapter 5	PRE Program Debugging and Compiling of Embedded Mode	41
5.1	PRE Debugging Instruction	42
5.2	PRE Compilation Instruction	43
Chapter 6	Debugging.....	46
6.1	Command Mode Debugging	47
6.1.1	Basic Command Mode.....	47
6.1.2	Advanced Command Mode	49
6.2	Embedded Mode Debugging.....	52

Chapter 7 Error Code Explanation 54

Figure Contents

Figure 2.1 Software Structure Diagram of Command Mode	5
Figure 3.1 Initialization Process	9
Figure 3.2 Card Writing Process	10
Figure 3.3 Exit Process	11
Figure 3.4 Initialization Process	12
Figure 3.5 Card Writing Process	13
Figure 3.6 Exit Process	14
Figure 4.1 Universal Interface Call Flow Chart of Command Mode	32
Figure 4.2 14443A Command Interface Call Flow Chart of Command Mode	33
Figure 4.3 14443B Command Interface Call Flow Chart of Command Mode	34
Figure 4.4 MIFARE Authentication Interface Call Flow Chart of Command Mode	35
Figure 4.5 Interface Call Flow Chart of Embedded Mode	36
Figure 4.6 libpt_card.so Interface Call Flow Chart of Embedded Mode	40
Figure 5.1 PRE Debugging Interface	42
Figure 6.1 Debugging of Basic Command Mode	47
Figure 6.2 Debugging of Advanced Command Mode	49
Figure 6.3 MIFARE Authentication of Advanced Command Mode	50
Figure 6.4 Embedded Mode Debugging	52
Figure 6.5 Script Operation Interface of Embedded Mode	53

Figure 6.6 Delete Script of Embedded Mode..... 53

Table Contents

Table 1.1 Glossary.....	2
Table 1.2 SDK Directory Structure	2
Table 4.1 Card Mode Description Table.....	18
Table 4.2 Card Communication Rate Description Table	19
Table 4.3 Card Mode Description Table	27
Table 5.1 Cross-compilation Environment	43
Table 5.2 Compile Template	44
Table 7.1 Error Code Explanation	55

Chapter 1 Introduction



1.1 Purpose

This document describes the software structure and principle of the PIOTEC contactless reader, which is used to guide the user software developer to carry out secondary development of the PIOTEC contactless reader, and then drive the PIOTEC contactless reader installed on a smart card personalization device to complete the personalized data input into cards.

1.2 Applicable Production

This manual applies to the PT1301N contactless reader developed and produced by PIOTEC.

1.3 Glossary

Table 1.1 Glossary

No.	Glossary	Abbreviation	Explanations
1	Personalized Script	Script	Store instruction files that conform to ISO14443A/B, ISO15693, MIFARE, and FELICA specifications.

1.4 SDK Directory Structure

Table 1.2 SDK Directory Structure

No.	Directory Name	Explanations
1	doc	Store the PIOTEC reader drive interface development manual.
2	PTCtlsReaderTest	The pt_card.dll program debugging project of secondary development by users.
3	PRE Compiler template	The PRE program compilation template project of secondary development by users.
4	PREDebug	The PRE program debugging project of secondary development by users.

5	Reader Lib	The related library files of secondary development by users.
6	bin	PTCtIsReaderTest debugger directory of secondary development by users.

Chapter 2 Software Structure



2.1 Software Structure of Command Mode

In command mode, the user needs to perform secondary development for components integrated in the user data management system, as shown in the orange part of Figure 2.1.

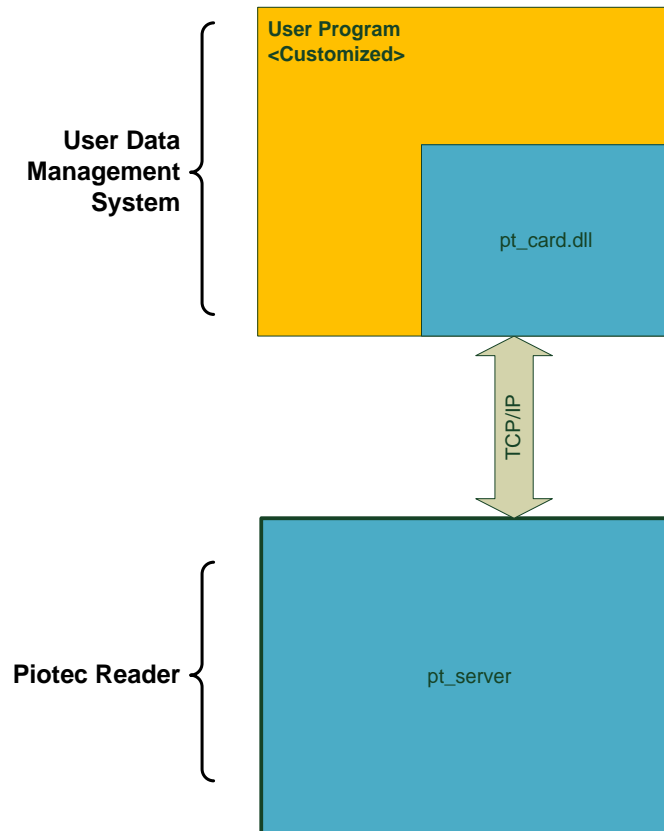


Figure 2.1 Software Structure Diagram of Command Mode

The system components in command mode are described as follows:

- `pt_card.dll`: The control module of Piotec reader, which does the interaction of command and data with Piotec readers through the network communication protocol to achieve various operations on the Piotec reader.
- `pt_server`: The reader service program, which completes the network communication with the reader control module and the internal program of the reader.

2.2 Software Structure of Embedded Mode

In embedded mode, the user needs to perform secondary development for two system components.

One is integrated in the user data management system, and the other is integrated in the Piotec reader, as shown in the orange part of Figure 2.2.

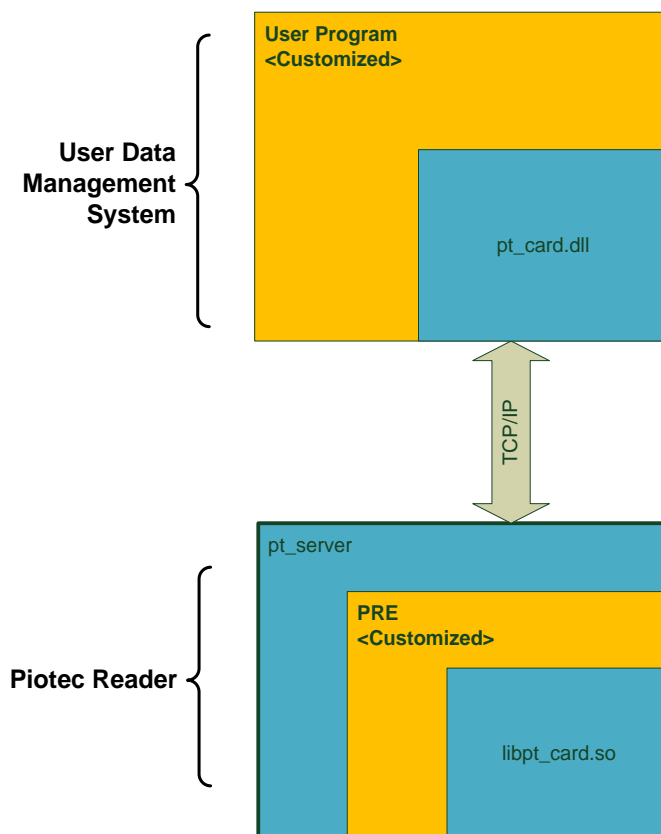


Figure 2.2 Software Structure Diagram of Embedded Mode

The system components in embedded mode are described as follows:

- **pt_card.dll**: The control module of Piotec reader, which does the interaction of command and data with Piotec readers through the network communication protocol to achieve various operations on the Piotec reader.
- **pt_server**: The reader service program, which completes the network communication with the reader control module, storage of internal files of the reader and calling 'PRE' program.
- **PRE**: An embedded write-card application that runs inside the Piotec reader and is used to execute write-card instructions. At the time of job initialization, the PRE program is downloaded by the data management system PT_PDM to the inside of the reader through the TCP/IP network protocol and

initiated by `pt_server`. The application allows users to do secondary development, and the customer can specify the name of the 'PRE' file as needed.

- `Lib libpt_card.so`: The reader driver, running inside the Piotec reader, provides a reader interface for the embedded write-card application 'PRE'.

Chapter 3 Software Workflow



3.1 Debugging of Command Mode

The reader's workflow consists of three parts: initialization, running, and exit.

- Initialization process: The user program completes the connection of the reader by calling the interface function of pt_card.dll, as shown in Figure 3.1.

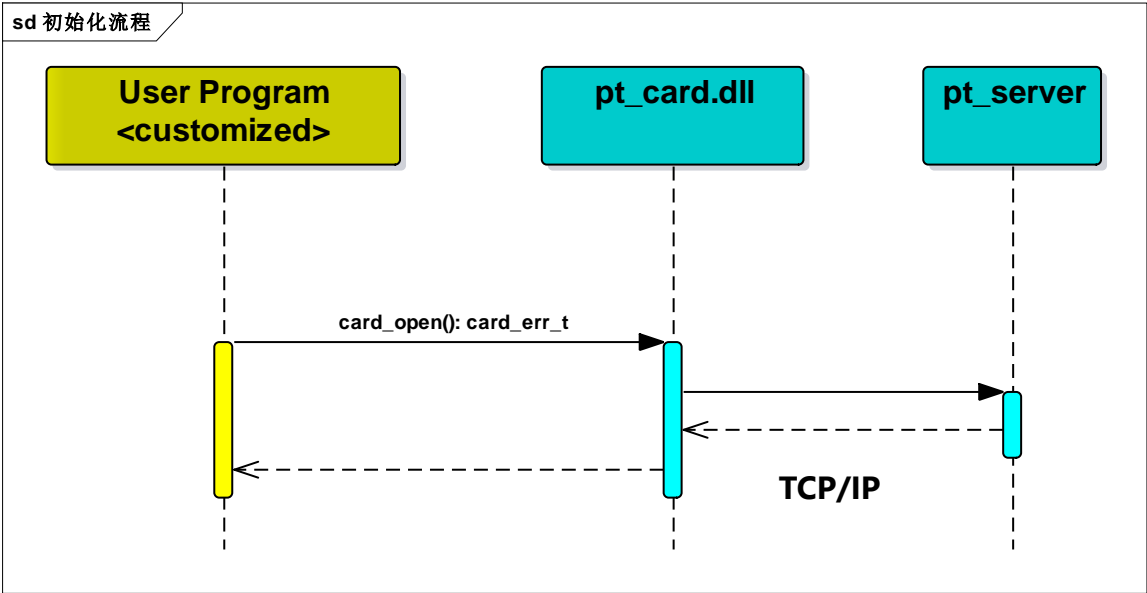


Figure 3.1 Initialization Process

- Running process: The user program completes the personalized writing of the chip by calling the interface function of pt_card.dll, as shown in Figure 3.2.

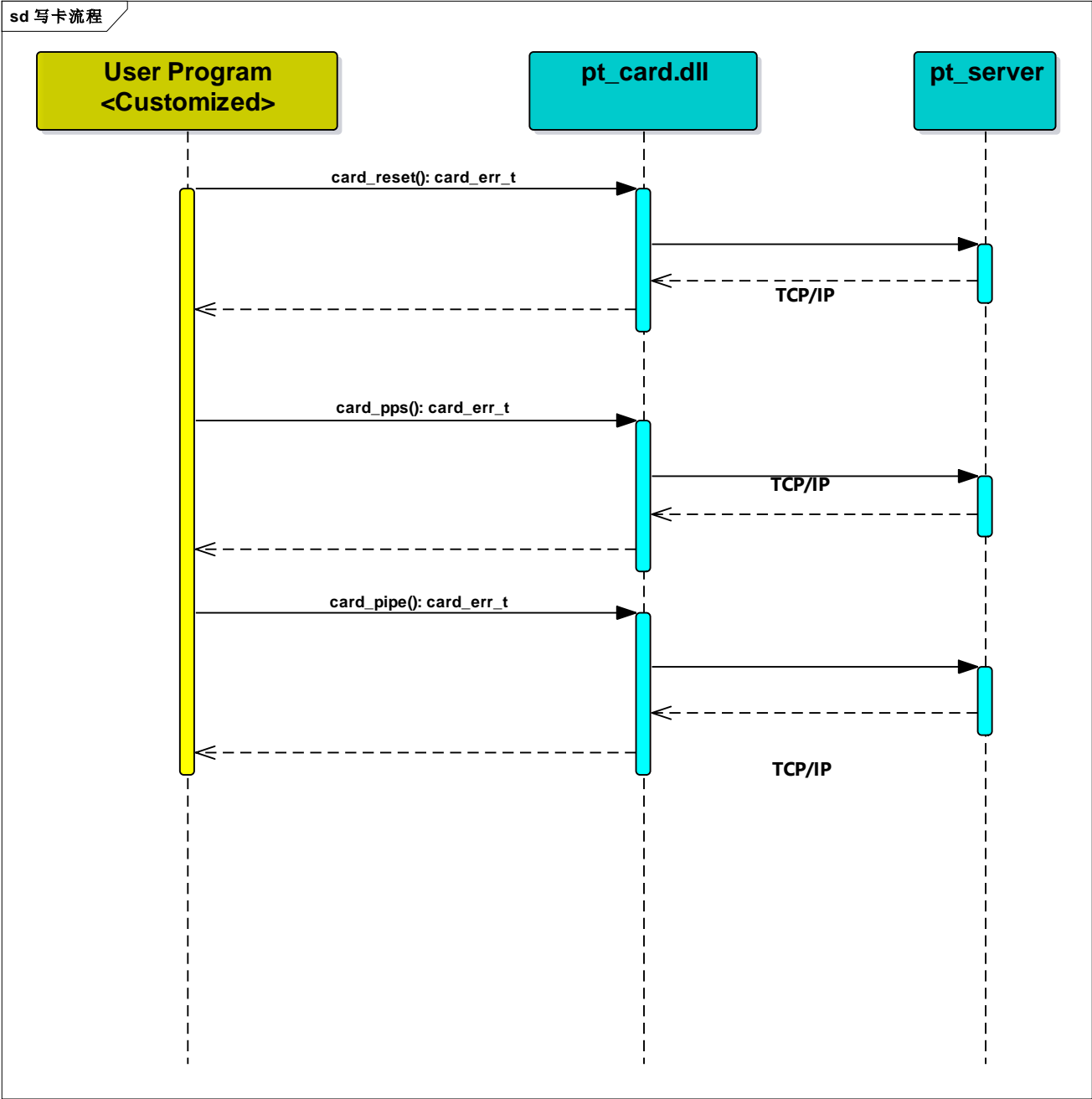


Figure 3.2 Card Writing Process

- Exit process: The user program completes the operation of disconnecting the reader by calling the interface function of pt_card.dll, as shown in Figure 3.3.

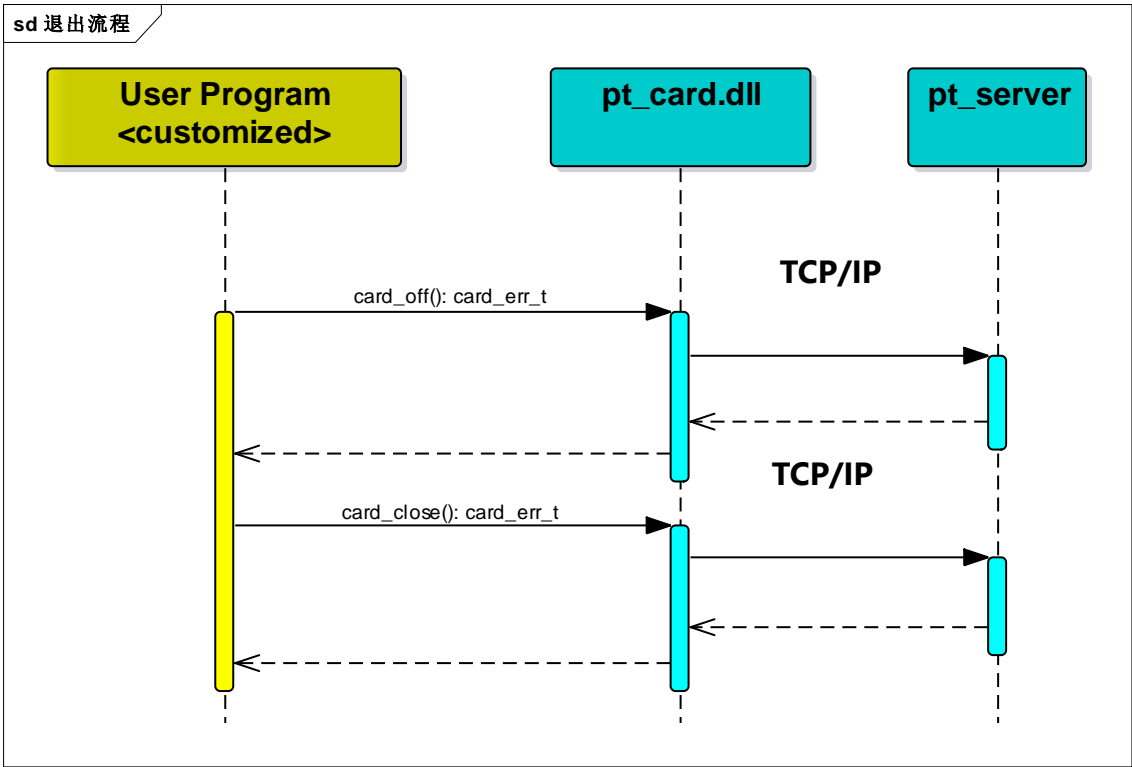


Figure 3.3 Exit Process

3.2 Debugging of Embedded Mode

The reader's workflow consists of three parts: initialization, running, and exit.

- Initialization process: The user program completes the connection of the reader by calling the interface function of pt_card.dll, as shown in Figure 3.4.

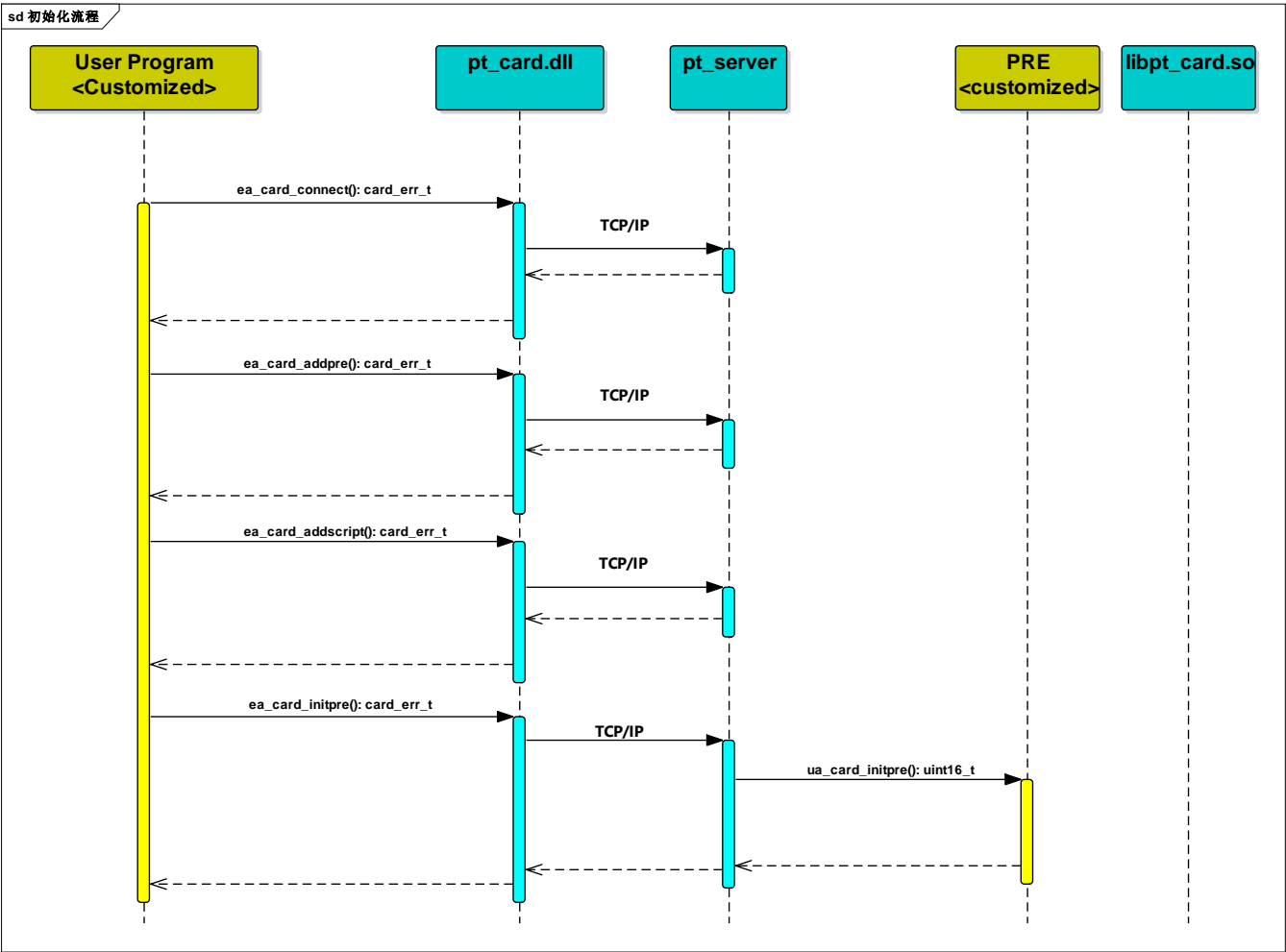


Figure 3.4 Initialization Process

- Running process: The user program completes the personalized writing of the chip by calling the interface function of pt_card.dll, as shown in Figure 3.5.

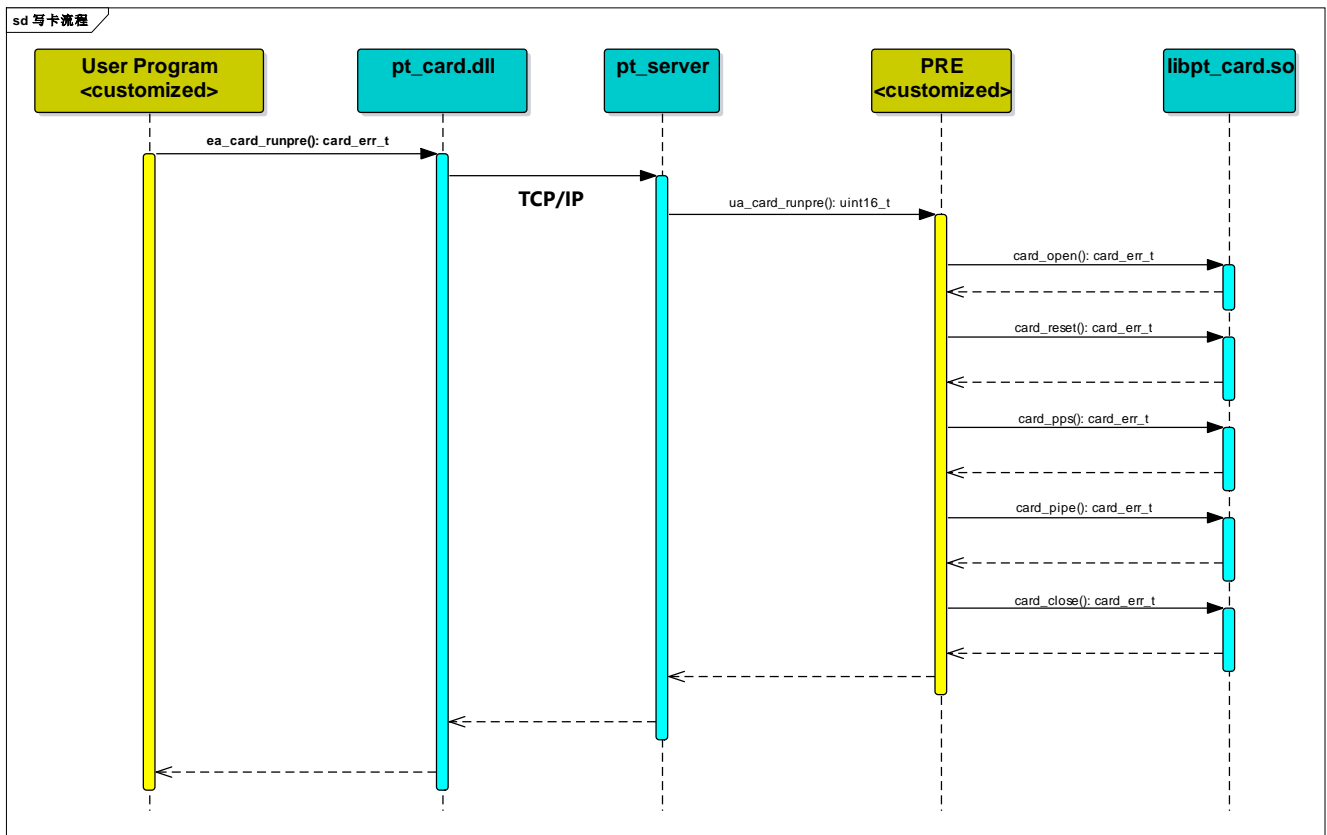


Figure 3.5 Card Writing Process

- Exit process: The user program completes the operation of disconnecting the reader by calling the interface function of pt_card.dll, as shown in Figure 3.6.

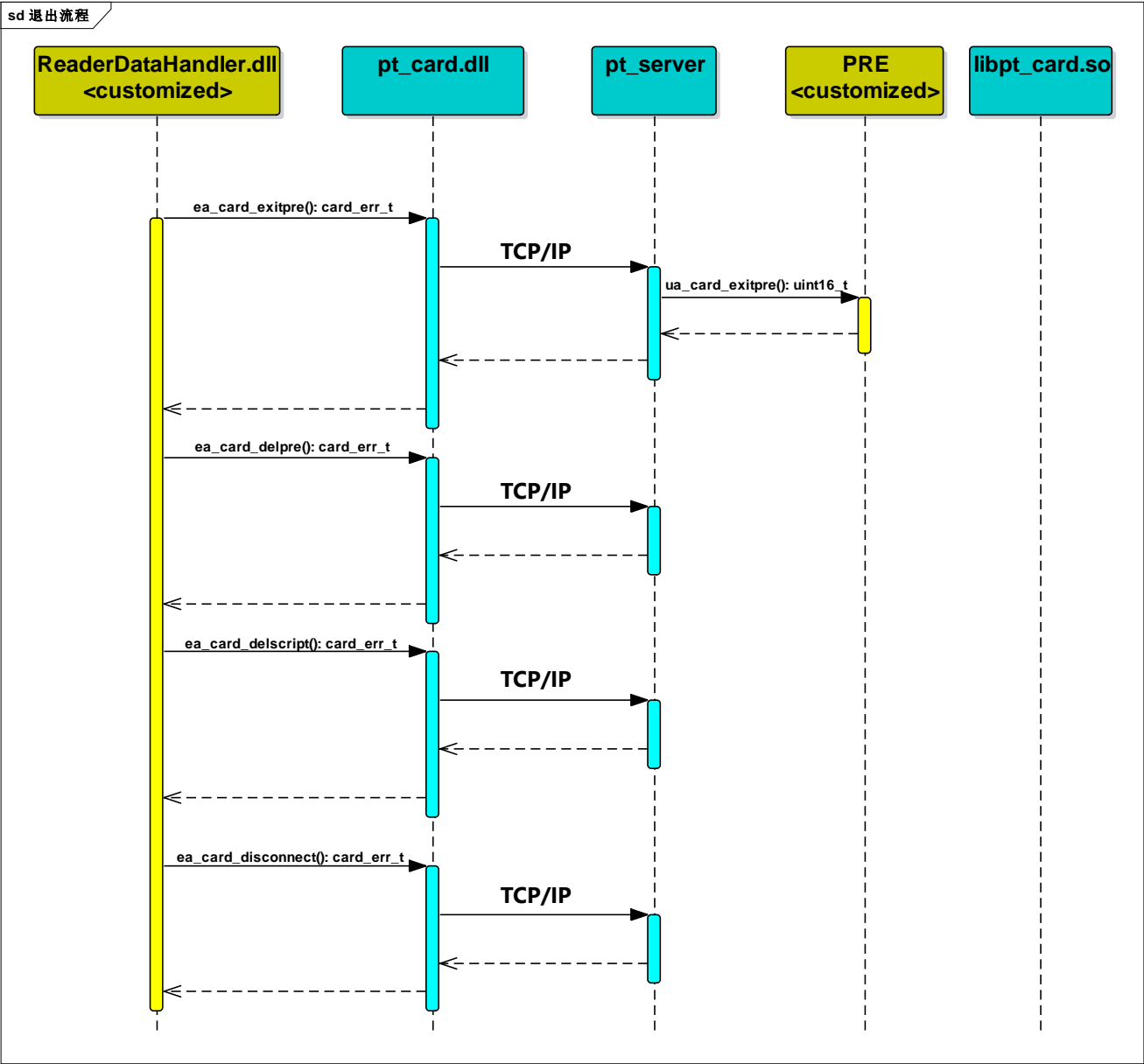


Figure 3.6 Exit Process

Chapter 4 Software Interface



Piotec provides the pt_card.dll driver interface to meet the user's secondary development. The user program utilizes the interface provided by pt_card.dll to implement the delivery of control commands and the personalization of smart card data.

4.1 Universal Interface of Command mode

4.1.1 Card object Structure

```
typedef struct card_object {
    Int32_t handle;                /* card object handle */
    Uint8_t addr[MAX_ADDR_SIZE];  /* card object address */
    card_mod_t model;              /* card mode */
    Uint8_t atr[ATR_DATA_LEN];    /* card ATR value */
    Uint8_t atr_len;               /* card ATR length */
    Uint8_t sw1;                   /* card return value SW1 */
    Uint8_t sw2;                   /* card return value SW2 */
    Uint32_t timeout;              /* communication timeout period */
    card_err_t last_err;           /* last error number */
} card_obj_t;
```

4.1.2 Contactless Card Configuration Structure

```
typedef struct card_picc_config {
    Uint32_t mask;                 /** tag mask */
    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;

```

This structure is currently not open for use.

4.1.3 card_open

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

Function: Initialized card communication object and communication mode.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **model[in]:** Card mode. Shown in Table 4.1.
- 3) **addr[in]:** Command mode: This parameter needs to be filled in the corresponding reader IP.

Embedded mode: When using the online debugging mode, this parameter needs to be filled in the corresponding reader IP, and it can be set as NULL during formal production.

Return value: Error code (return 0 if correct).

Table 4.1 Card Mode Description Table

Card Mode	Enumeration Definition of card_mode_t
ISO14443A	MODEL_P14443A
ISO14443B	MODEL_P14443B
MIFARE	MODEL_PMIFARE
FELICA	MODEL_PFELICA
ISO15693	MODEL_P15693

4.1.4 card_close

- card_err_tcard_close (card_obj_t *obj)

Function: Power off the card and close the reader to release resources.

Parameters:

- 1) **obj [in/out]:** Card object structure.

Return value: Error code (return 0 if correct).

4.1.5 card_reset

- card_err_tcard_reset (card_obj_t *obj)

Function: Turn on the RF carrier and perform a card reset.

Parameters:

- 1) **obj [in/out]:** Card object structure.

Return value: Error code (return 0 if correct).

4.1.6 card_on

- card_err_tcard_on (card_obj_t *obj)

Function: Turn on the reader RF carrier.

Parameters:

- 1) **obj [in/out]:** Card object structure.

Return value: Error code (return 0 if correct).

4.1.7 card_off

- `card_err_t card_off (card_obj_t *obj)`

Function: Turn off the reader RF carrier.

Parameters:

- 1) **obj [in/out]:** Card object structure.

Return value: Error code (return 0 if correct).

4.1.8 card_pcfg

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

Function: Contactless reader configuration settings. **Temporarily it is not open for use.**

Parameters:

- 1) **obj [in/out]:** Card object structure.
- 2) **pcfg [in]:** Contactless reader configuration structure.

Return value: Error code (return 0 if correct).

4.1.9 card_pps

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

Function: Set the card communication rate.

Parameters:

- 1) **obj [in/out]:** Card object structure.
- 2) **param1[in]:** DSI value.
- 3) **param2[in]:** DRI value.

Return value: Error code (return 0 if correct).

Table 4.2 Card Communication Rate Description Table

DSI DRI	Macro Definition
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_tcard_pipe (card_obj_t *obj, Uint8_t *tbuf, Uint16_t tlen, Uint8_t *rbuf, Uint16_t *rlen)

Function: The reader exchanges data with the card.

Parameters:

- 1) **obj [in/out]:** Card object structure.
- 2) **tbuf [in]:** Send data cache.
- 3) **tlen [in]:** Send data length.
- 4) **rbuf [out]:** Receive data cache.
- 5) **rlen [out]:** Receive data length.

Return value: Error code (return 0 if correct).

4.1.11 card_getinfo

- card_err_tcard_getinfo (card_obj_t *obj, Uint8_t *info)

Function: Obtain reader device information.

Parameters:

- 1) **obj [in/out]:** Card object structure.
- 2) **info[out]:** Device information, the character length is 64 bytes, and the current device information is the version number.

Return value: Error code (return 0 if correct).

4.1.12 card_setfreq

- card_err_tcard_setfreq (card_obj_t *obj, Uint16_t freq)

Function: Set the reader writing frequency. **Temporarily it is not open for use.**

Parameters:

- 1) **obj [in/out]:** Card object structure.
- 2) **freq[in]:** Card writing frequency. Range is 1000~20000. Unit is KHz.

Return value: Error code (return 0 if correct).

4.1.13 card_getfreq

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

Function: Obtain the reader card writing frequency value. **Temporarily it is not open for use.**

Parameters:

- 1) **obj [in/out]:** Card object structure.
- 2) **freq[out]:** Obtain the reader card writing frequency value. Unit is KHz. Range is 1000~20000.KHz. Default is 3570KHz.

Return value: Error code (return 0 if correct).

4.1.14 card_setmodel

- `card_err_t card_setmodel (card_obj_t *obj, card_mod_t model)`

Function: Set the card mode.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **model[in]:** Card mode. Shown in Table 4.1.

Return value: Error code (return 0 if correct).

4.1.15 card_getmodel

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

Function: Obtain the current card mode.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **model[out]:** Card mode. Shown in Table 4.1.

Return value: Error code (return 0 if correct).

4.1.16 card_automodel

- `card_err_t card_automodel (card_obj_t *obj)`

Function: Set the card mode automatically. Only 14443A, 14443B and MIFARE cards are supported.

Parameters:

- 1) **obj[in/out]:** Card object structure.

Return value: Error code (return 0 if correct).

4.2 ISO14443A Command Interface of Command Mode

4.2.1 card_reqa

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

Function: The Type A card sends a request command, which is used after the card_on carrier is turned on.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **atqa[out]:** Type A card request response, fixed 2 bytes in length.

Return value: Error code (return 0 if correct).

4.2.2 card_wupa

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

Function: The Type A card sends a wake-up command that is used after calling the card_halta stop state.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **atqa[out]:** Type A card request response, fixed 2 bytes in length.

Return value: Error code (return 0 if correct).

4.2.3 card_halta

- `card_err_t card_halta(card_obj_t *obj)`

Function: Execute the command of stopping type A card. The card can only be woken up by

card_wupa.

Parameters:

- 1) **obj[in/out]:** Card object structure.

Return value: Error code (return 0 if correct).

4.2.4 card_select

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

Function: Execute the command of selecting cards to select a card with a known UID.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **uid[in]:** Card unique identifier.
- 3) **uid_len[in]:** The length of the card unique identifier.
- 4) **sak[out]:** Select acknowledge.

Return value: Error code (return 0 if correct).

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)`

Function: Execute the anti-collision command and select the card. The anti-collision command must be used after card_reqa or card_wupa.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **uid[in]:** Card unique identifier.
- 3) **uid_len[in]:** The length of the card unique identifier.
- 4) **sak[out]:** Select acknowledge.
- 5) **status[out]:** Status flag, 0: no collision detected, 1: collision detected.

Return value: Error code (return 0 if correct).

4.2.6 card_rats

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

Function: Execute the command of card request response selection to activate the ISO14443-4 layer protocol.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **ats[out]:** The card requests a response selection.
- 3) **ats_len[out]:** The length of the card requests a response selection.

Return value: Error code (return 0 if correct).

4.3 ISO14443B Command Interface of Command Mode

4.3.1 card_reqb

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

Function: The B-type card sends a request command, which is used after the card_on carrier is turned on.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **atqb[out]:** Type B card request response.
- 3) **atqb_len[out]:** The length of type B card request response.

Return value: Error code (return 0 if correct).

4.3.2 card_wupb

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

Function: The B-type card sends a wake-up command, which is used to call the card_haltb stop state.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **atqb[out]:** Type B card request response.
- 3) **atqb_len[out]:** The length of type B card request response.

Return value: Error code (return 0 if correct).

4.3.3 card_haltb

- `card_err_t card_haltb(card_obj_t *obj)`

Function: Execute the command of stopping type B card. The card can only be woken up by `card_wupa`.

Parameters:

- 1) **obj[in/out]:** Card object structure.

Return value: Error code (return 0 if correct).

4.3.4 card_attrib

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

Function: Execute the type B card attrib command.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **rbuf[out]:** Attrib command response data.
- 3) **rlen[out]:** The length of attrib command response data.

Return value: Error code (return 0 if correct).

4.3.5 card_setattribinf

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

Function: Set high-level configuration of the type B card attrib command.

Parameters:

- 1) **obj[in/out]:** Card object structure.
- 2) **tbuf[in]:** High-level configuration information in Attrib command.
- 3) **tlen[in]:** The length of high-level configuration information in Attrib command.

Return value: Error code (return 0 if correct).

4.3.6 card_getattribinf

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

Function: Obtain high-level configuration of the type B card attrib command.

Parameters:

- 1) **obj[in/out]**: Card object structure.
- 2) **rbuf[out]**: High-level configuration information in Attrib command.
- 3) **rlen[out]**: The length of high-level configuration information in Attrib command.

Return value: Error code (return 0 if correct).

4.4 ISO14443 Universal Command Interface of Command mode

4.4.1 card_getid

- `card_err_tcard_getid(card_obj_t *obj, Uint8_t *id, Uint8_t *id_len)`

Function: Obtain the card identifier.

Parameters:

- 1) **obj[in/out]**: Card object structure.
- 2) **id[out]**: Card identifier. ISO14443A: UID、ISO14443B: PUPI.
- 3) **id_len[out]**: The length of card identifier.

Return value: Error code (return 0 if correct).

4.4.2 card_deselect

- `card_err_tcard_deselect(card_obj_t *obj)`

Function: Send an S command to deactivate the card. Disable the 14443-4 layer protocol. The reader is in the 14443-3 layer protocol. If the execution is successful, the card should be in a stopped state and can only be activated by card_wupa.

Parameters:

- 1) **obj[in/out]**: Card object structure.

Return value: Error code (return 0 if correct).

4.4.3 card_getdetect

- `card_err_tcard_getdetect(card_obj_t *obj, Uint8_t *val)`

Function: Check for the presence of cards inside the antenna. The reader R (NAK) block sends, the card R (ACK) block responses. The card and reader are required to be in the 14443-4 layer protocol.

Parameters:

- 1) **obj[in/out]**: Card object structure.
- 2) **val[out]**: 1 byte, 1: card exists, 0: card does not exist.

Return value: Error code (return 0 if correct).

4.5 MIFARE Authentication Interface of Command Mode

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)`

Function: MIFARE card authentication.

Parameters:

- 1) **obj[in/out]**: Card object structure.
- 2) **block_no[in]**: Block number.
- 3) **key_type[in]**: Card authentication key type.
- 4) **key[in]**: Card key. The key is 6 bytes in length.
- 5) **uid[in]**: Card UID.

Return value: Error code (return 0 if correct).

Table 4.3 Card Mode Description Table

MIFARE Card Key Type	Macro Definition
Key A	CARD_MIFARE_KEYA
Key B	CARD_MIFARE_KEYB

4.6 Embedded Mode Interface

4.6.1 ea_card_connect

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

Function: Connect the reader.

Parameters:

- 1) **obj [in/out]**: Card object structure.
- 2) **addr[in]**: This parameter needs to be filled in the corresponding reader IP.

Return value: Error code (return 0 if correct).

4.6.2 ea_card_disconnect

- `card_err_tea_card_disconnect(card_obj_t *obj);`

Function: Disconnect the reader.

Parameters:

- 1) **obj [in/out]:** Card object structure.

Return value: Error code (return 0 if correct).

4.6.3 ea_card_addpre

- `card_err_tea_card_addpre(card_obj_t *obj, Uint8_t *path, Uint8_t *name, Uint8_t *md5, Uint8_t force);;`

Function: Download the PRE program to the reader.

Parameters:

- 1) **obj [in/out]:** Card object structure.
- 2) **path[in]:** The path and program name of PRE program stored in host computer.
- 3) **name[in]:** The name of the PRE program inside the reader.
- 4) **md5[out]:** Returns MD5 value of PRE file after download successfully (length in 16 bytes).
- 5) **force[in]:** Overwriting. 0-Check if the file exists. 1-do not check if the file exists, overwriting.

Return value: Error code (return 0 if correct).

4.6.4 ea_card_addscript

- `card_err_tea_card_addscript(card_obj_t *obj, Uint8_t *path, Uint8_t *name, Uint8_t *md5, Uint8_t force);`

Function: Download the script file to the reader.

Parameters:

- 1) **obj [in/out]:** Card object structure.
- 2) **path[in]:** The path and file name of the script file stored in host computer.
- 3) **name[in]:** The script file storage name inside the reader.
- 4) **md5[out]:** Returns MD5 value of PRE file after download successfully (length in 16 bytes).
- 5) **force[in]:** Overwriting. 0-Check if the file exists. 1-do not check if the file exists, overwriting.

Return value: Error code (return 0 if correct).

4.6.5 ea_card_delpre

- `card_err_tea_card_delpre(card_obj_t *obj, Uint8_t *name);`

Function: Delete the PRE program stored in the reader.

Parameters:

- 1) ***obj [in/out]:*** Card object structure.
- 2) ***name[in]:*** The name of the PRE program stored inside the reader.

Return value: Error code (return 0 if correct).

4.6.6 ea_card_delscript

- `card_err_tea_card_delscript(card_obj_t *obj, Uint8_t *name);`

Function: Delete the script file stored in the reader.

Parameters:

- 1) ***obj [in/out]:*** Card object structure.
- 2) ***name[in]:*** The name of the script file stored inside the reader.

Return value: Error code (return 0 if correct).

4.6.7 ea_card_listpre

- `card_err_tea_card_listpre(card_obj_t *obj, Uint8_t *list, Uint32_t list_len, Uint32_t *list_all_len);`

Function: List all PRE programs stored in the reader.

Parameters:

- 1) ***obj [in/out]:*** Card object structure.
- 2) ***list[in]:*** Collection cache of PRE program name. Program names are separated by commas, such as "name1, name2".
- 3) ***list_len[in]:*** The length of collection cache of PRE program name.
- 4) ***list_all_len[out]:*** The actual length of collection cache of PRE program name.

Return value: Error code (return 0 if correct).

4.6.8 ea_card_listscript

- `card_err_tea_card_listscript(card_obj_t *obj, Uint8_t *list, Uint32_t list_len, Uint32_t`

*list_all_len);

Function: List all script files stored in the reader.

Parameters:

- 1) **obj [in/out]:** Card object structure.
- 2) **list[in]:** Collection cache of script file names. File names are separated by commas, such as "script 1, script 2".
- 3) **list_len[in]:** The length of the script files name collection.
- 4) **list_all_len[out]:** The actual length of the script files name collection.

Return value: Error code (return 0 if correct).

4.6.9 ea_card_initpre

- `card_err_tea_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);`

Function: Initialize the PRE program.

Parameters:

- 1) **obj [in/out]:** Card object structure.
- 2) **pre_name[in]:** The name of the PRE program inside the reader.
- 3) **script_name[in]:** The name of the script file inside the reader.
- 4) **user_data[in]:** User-defined input data, such as keyword information in scripts.
- 5) **user_data_len[in]:** User-defined input data length.
- 6) **output[out]:** User-defined output data, such as error messages.
- 7) **output_info_len[in/out]:** User-defined output data length. You need to input the output to open the buffer length and output the actual output data length.

Return value: Error code (return 0 if correct).

4.6.10 ea_card_runpre

- `card_err_tea_card_runpre(card_obj_t *obj, Uint8_t *user_data, Uint32_t user_data_len, Uint8_t *output_info, Uint32_t *output_info_len);`

Function: Run the PRE program once.

Parameters:

- 1) ***obj [in/out]***: Card object structure.
- 2) ***user_data [in]***: User-defined input data, such as personalized write data.
- 3) ***user_data_len [in]***: User-defined input data length.
- 4) ***output[out]***: User-defined output information, such as error messages.
- 5) ***output_info_len[in/out]***: User-defined output data length. You need to input the output to open the buffer length and output the actual output data length.

Return value: Error code (return 0 if correct).

4.6.11 ea_card_exitpre

- `card_err_tea_card_exitpre(card_obj_t *obj, Uint8_t *output_info, Uint32_t *output_info_len);`

Function: End the PRE program.

Parameters:

- 1) ***obj [in/out]***: Card object structure.
- 2) ***output[out]***: User-defined output information, such as error messages.
- 3) ***output_info_len[in/out]***: User-defined output data length. You need to input the output to open the buffer length and output the actual output data length.

Return value: Error code (return 0 if correct).

4.7 Interface Call Flow Chart of Command Mode

4.7.1 Universal Interface Call Flow Chart

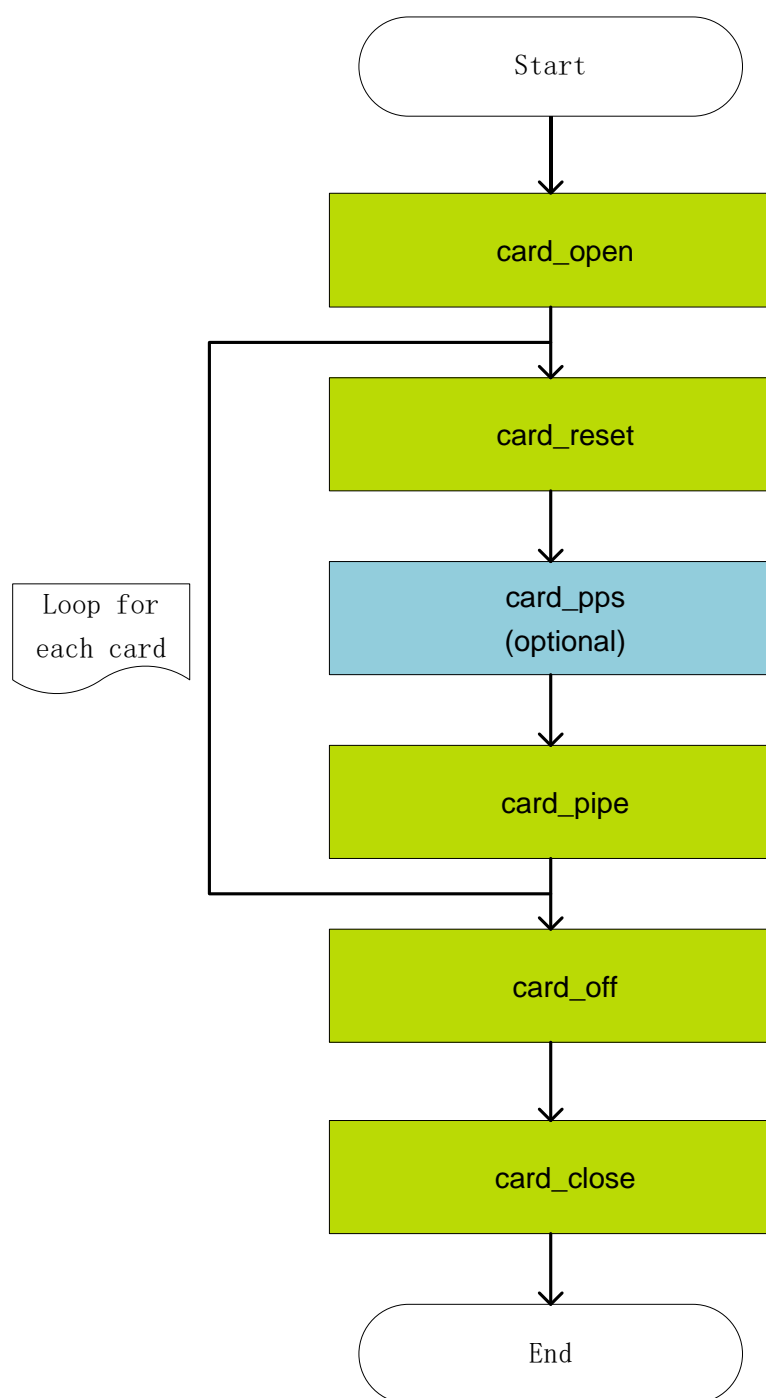


Figure 4.1 Universal Interface Call Flow Chart of Command Mode

4.7.2 ISO14443A Command Interface

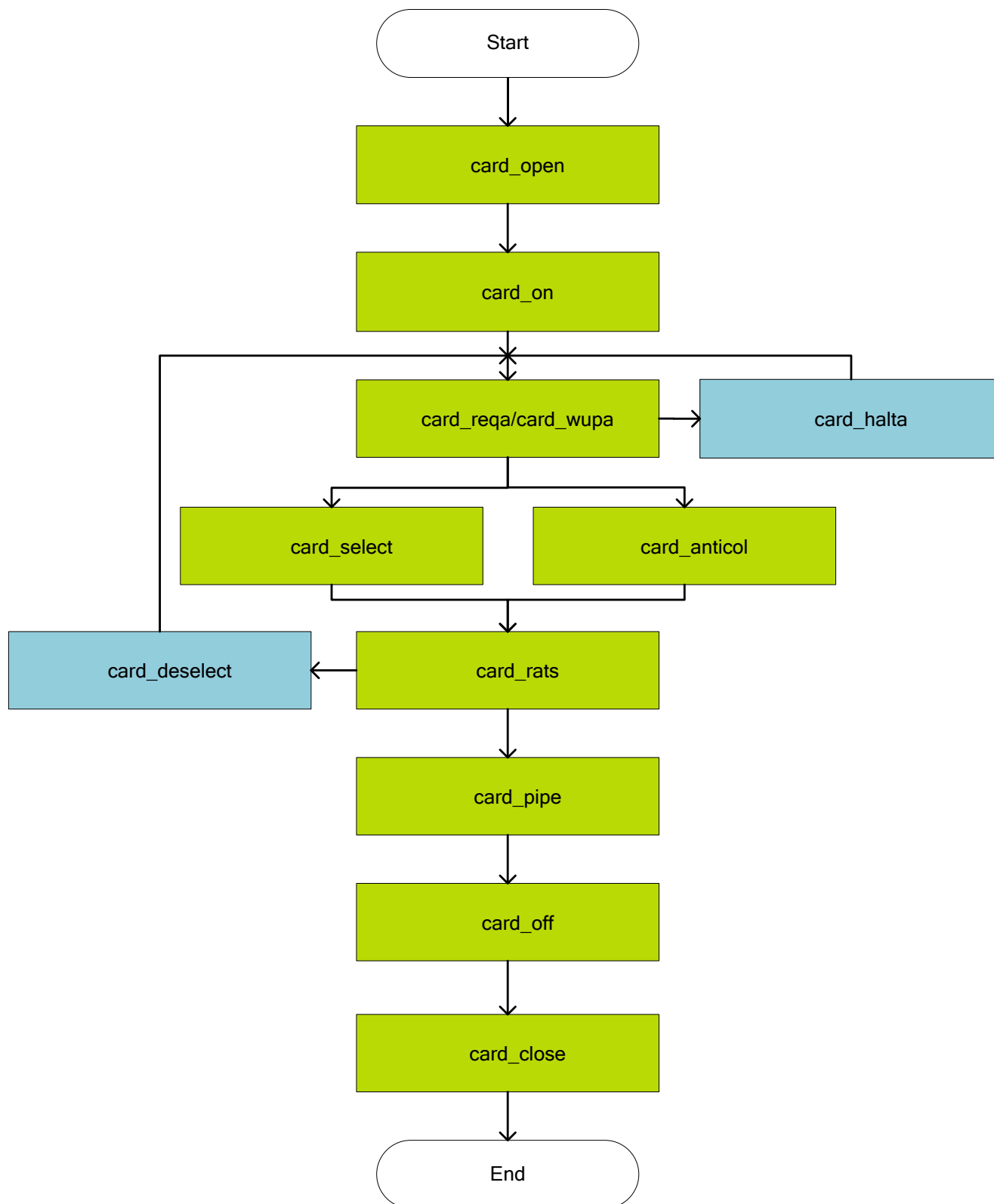


Figure 4.2 14443A Command Interface Call Flow Chart of Command Mode

4.7.3 ISO14443B Command Interface

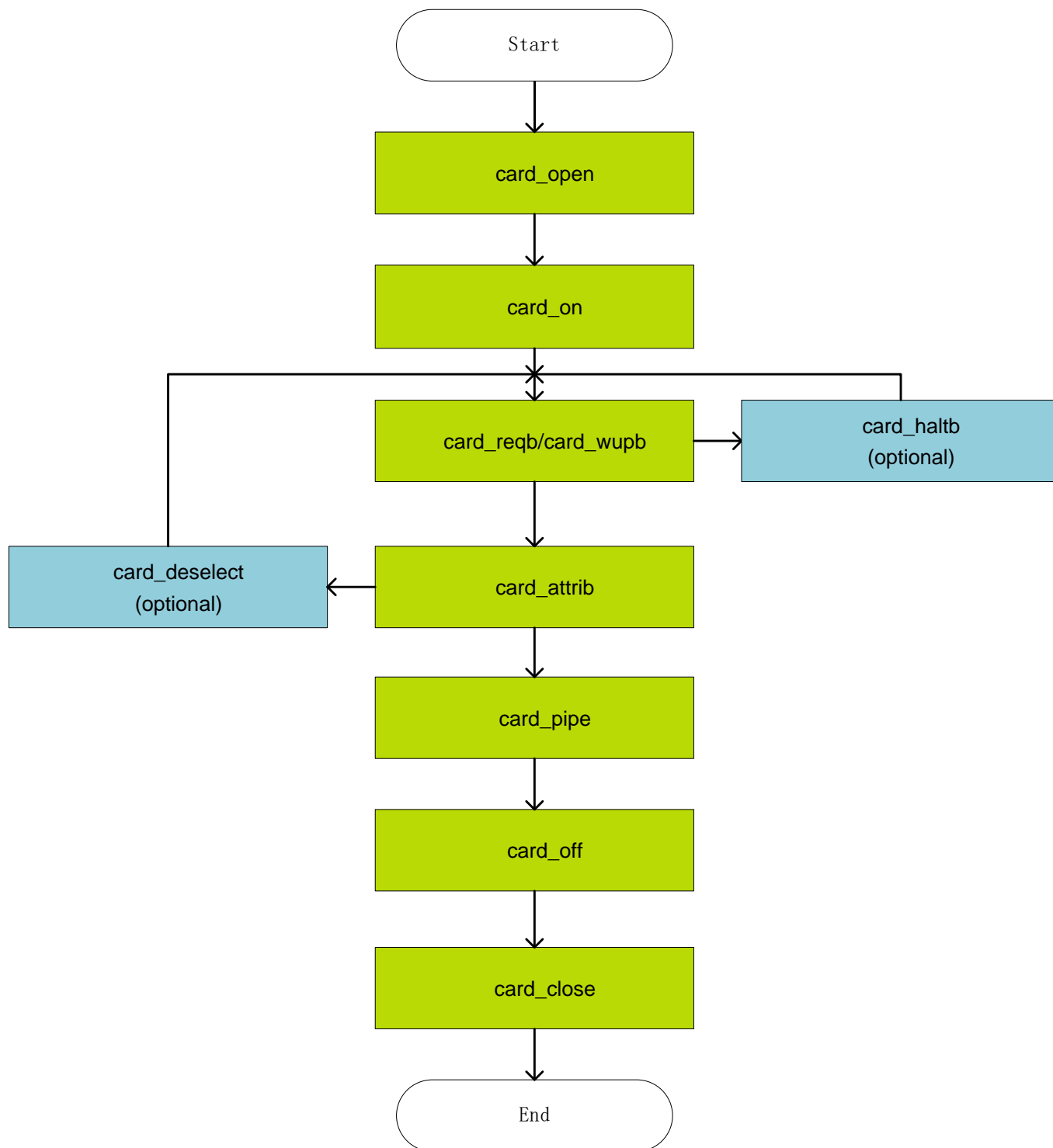


Figure 4.3 14443B Command Interface Call Flow Chart of Command Mode

4.7.4 MIFARE Authentication Interface

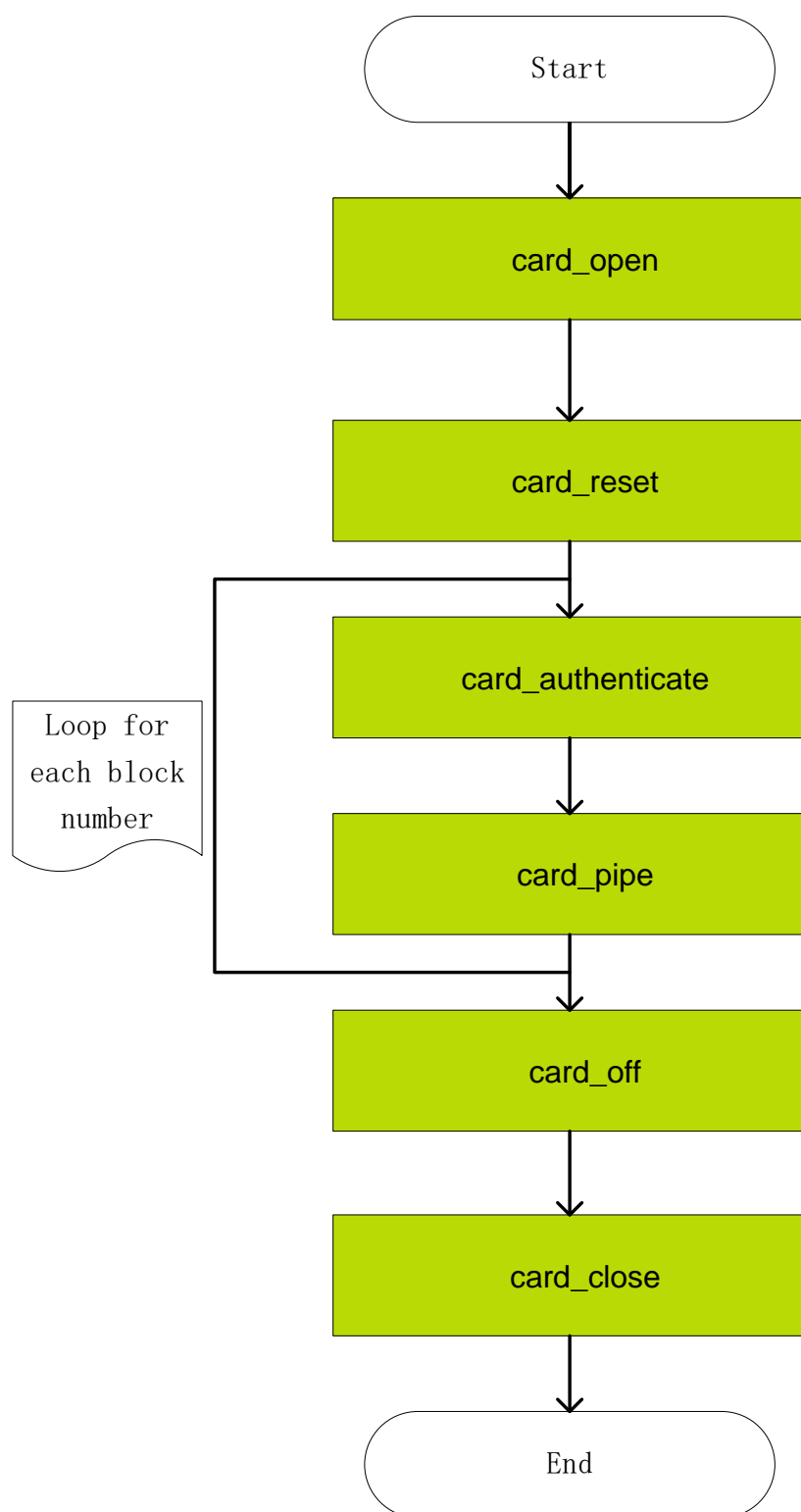


Figure 4.4 MIFARE Authentication Interface Call Flow Chart of Command Mode

4.8 Interface Call Flow Chart of Embedded Mode

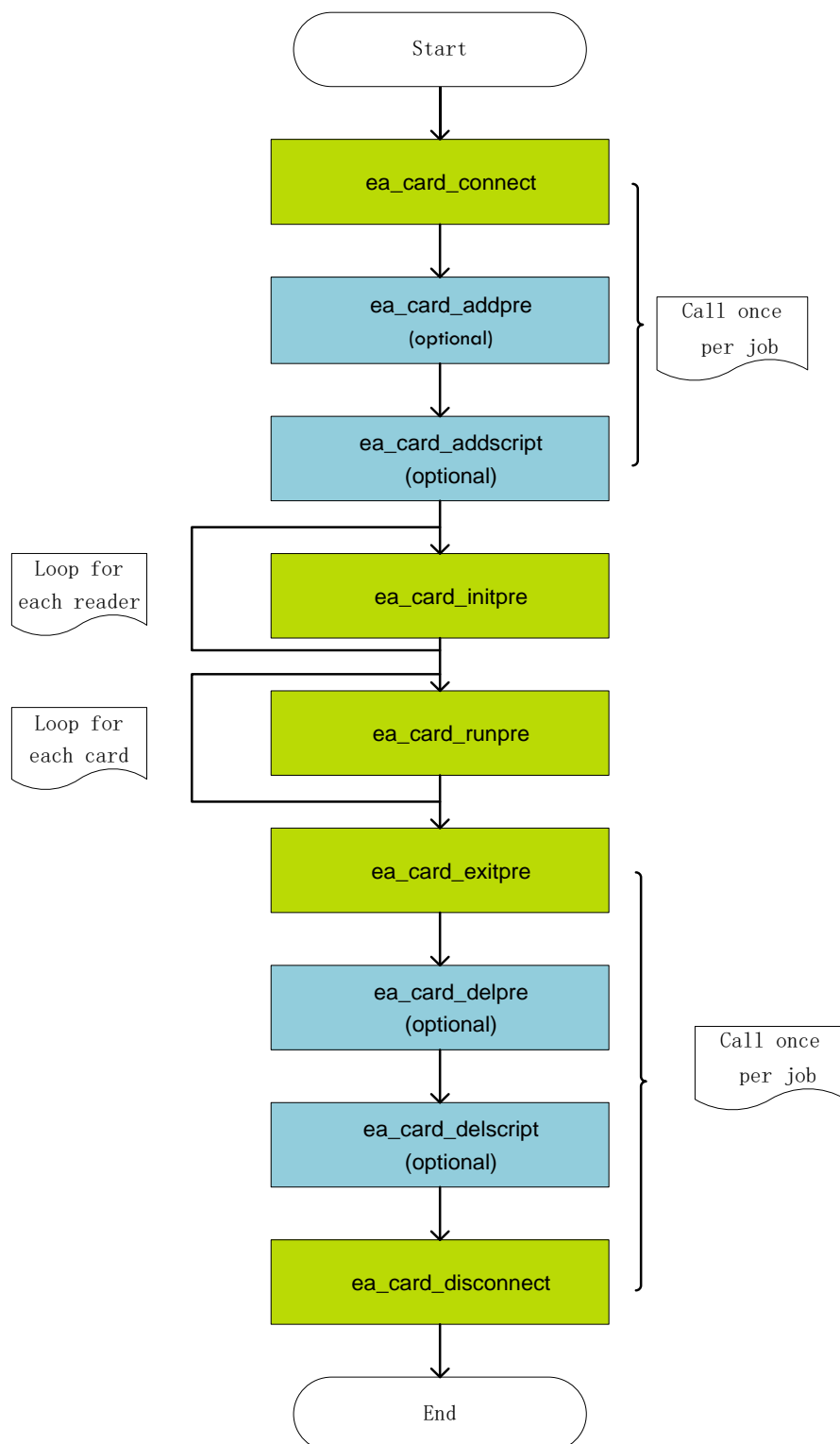


Figure 4.5 Interface Call Flow Chart of Embedded Mode

4.9 PRE Program Interface of Embedded Mode

The PRE program runs inside the reader and the program needs to be developed by using the intrinsic function interface template. The function of the interface function needs to be developed by the user.

<Template Program>

```
#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;

}

#endif

#endif
```

4.9.1 ua_card_initpre

- `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)`

Function: Initialize the PRE.

Parameters:

- 1) **script_path [in]:** The path and name of the script file stored inside the reader.
- 2) **user_data [in]:** User-defined input data, such as keyword information in scripts.
- 3) **user_data_len [in]:** User-defined input data length.
- 4) **output_info [out]:** User-defined output information, such as error messages.
- 5) **output_info_len [in/out]:** User-defined output information length. Length is not allowed to exceed the input length.

Return value: Error code (return 0 if correct).

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)`

Function: Run the PRE.

Parameters:

- 1) **user_data [in]:** User-defined input data, such as personalized write data.
- 2) **user_data_len [in]:** User-defined input data length.
- 3) **output_info [out]:** User-defined output information, such as error messages.
- 4) **output_info_len [in/out]:** User-defined output information length. Length is not allowed to exceed the input length.

Return value: Error code (return 0 if correct).

4.9.3 ua_card_exitpre

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

Function: Exit the PRE.

Parameters:

- 1) **output_info [out]**: User-defined output information, such as error messages.
- 2) **output_info_len [in/out]**: User-defined output information length. Length is not allowed to exceed the input length.

Return value: Error code (return 0 if correct). User-definable (0x5000 or above).

4.10 libpt_card.so Universal Interface Call Flow Chart of Embedded Mode

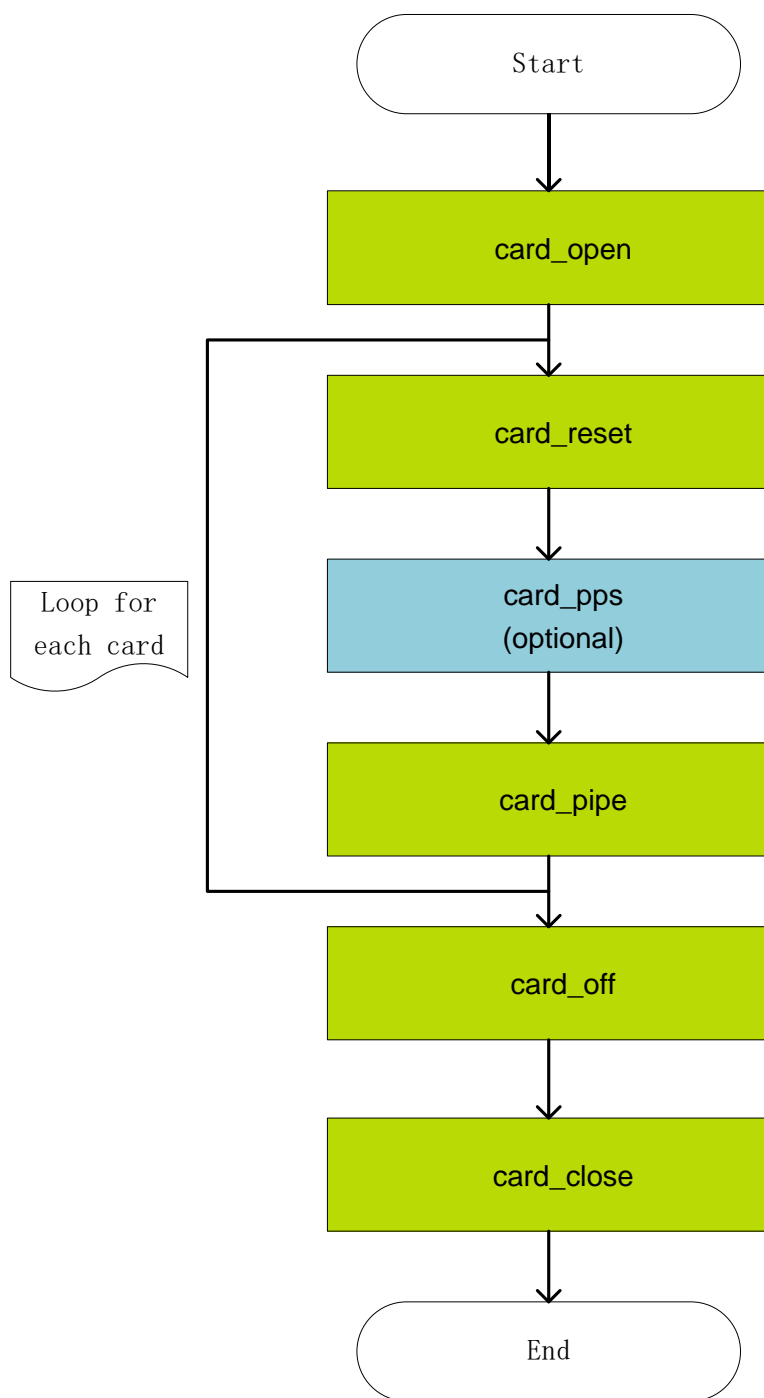



Figure 4.6 libpt_card.so Interface Call Flow Chart of Embedded Mode

Chapter 5 PRE Program Debugging and Compiling of Embedded Mode



5.1 PRE Debugging Instruction

The PRE program secondarily developed by the user can be tested using the "**Debug Project PREDebug**" (The storage path is: \\Piotec reader SDK\ PREDebug) provided by PIOTEC.

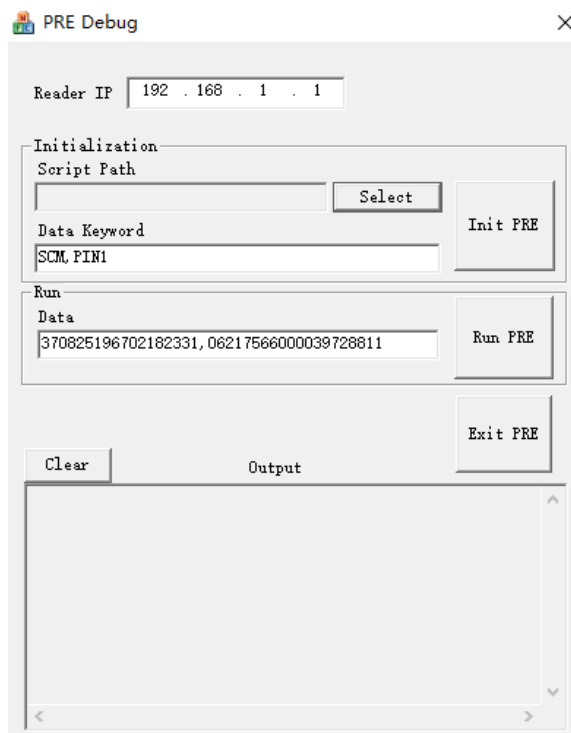


Figure 5.1 PRE Debugging Interface

The test process is as follows:

- 1) Use a network cable to connect the test PC to the reader hardware.
- 2) Insert the completed PRE program source code into the "**Debug Project PREDebug**".
- 3) Run the PREDebug program. The interface is shown in Figure 5.1.
- 4) Enter the IP address of the test reader in "**Reader IP**" in the interface shown in Figure 5.1. The default IP address is "192.168.1.1".
- 5) Init PRE([ua_card_initpre](#) interface function) test:
 - Click the 'Select' button to select the script path you want to pass in. This path will be passed to the 'script_path' parameter of the [ua_card_initpre](#) function.
 - Enter the data variable name in the 'Data Keyword' text box. The name will be passed to the 'user_data' parameter of the [ua_card_initpre](#) function.

- Click the 'Init PRE' button to test the call to the `ua_card_initpre` interface function.
- 6) Run PRE(`ua_card_runpre` interface function) test:
 - Enter the personalized data in the 'Data' text box, which will be passed to the '`user_data`' parameter of the `ua_card_runpre` function.
 - Click the 'Run PRE' button to test the call to the `ua_card_runpre` interface function.
- 7) Exit PRE(`ua_card_exitpre` interface function) test:
 - Click the 'Exit PRE' button to test the call to the `ua_card_exitpre` interface function.
- 8) The test results can be viewed through the list box below the interface.

5.2 PRE Compilation Instruction

The PRE program is compiled by using the compiled template project PRE Compiler template provided by PIOTEC. The compilation environment is a Linux platform, and an ARM-LINUX cross-compilation environment needs to be installed.

Table 5.1 Cross-compilation Environment

Cross-compilation Environment	Explanation
Installation Environment	Ubuntu 12.04.5、Virtualbox
Installation package	4.3.3.tgz
Compiler version	gcc version 4.3.3 (Sourcery G++ Lite 2009q1-176)

The process of installing a cross-compilation environment is as follows:

Direct import method:

- 1) Install Virtualbox and import the configured Ubuntu 12.04.5 image file `piotec.ova`.

Reinstallation method:

- 1) Install Virtualbox and install Ubuntu 12.04.5 system.
- 2) Log in to the Ubuntu system, copy the cross-editor installation package `4.3.3.tgz` to `/usr/local`, and extract the installation package to this folder.

Related commands:

```
cp toolchain.tar.gz /usr/local
```

```
cd /usr/local
```

```
tar -xzvf toolchain.tar.gz
```

3) Configure environment variables to take effect.

Related commands:

```
vim /etc/bash.bashrc
```

```
//Add at the end
```

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

```
// Save and exit
```

```
Enter:wq
```

```
//Make it effective immediately
```

```
source /etc/bash.bashrc
```

4)]Enter arm- linux- gcc-v and display the version number: gcc version 4.3.3 (Sourcery G++ Lite 2009q1-203), indicating that the installation was successful.

Compile template project PRE Compiler template (The storage path is: \\Piotec reader SDK\ PRE Compiler template) the project contains the following files and directories:

Table 5.2 Compile Template

Directory and File Name	Explanation
include directory	Contains the pt_card.h file required for the call
lib directory	Contains the plibpt_card.so file required for the call
PRE directory	Stores the compiled user PRE program
src directory	Stores user PRE program source code he directory contains the template source file sample.cpp
Makefile directory	Execute the compiled file

The compilation process is as follows:

- 1) Store the debugged user's PRE program source code in the src directory, for example: sample.cpp in the src directory.
- 2) Execute "**make SRC=user PRE source file name**" in the project directory, for example: make SRC=sample.
- 3) If the compilation is successful, the user PRE program will be automatically generated in the PRE directory, for example: sample.pre in the PRE directory.

Chapter 6 Debugging



During the secondary development, the user can test the process of calling pt_card.dll using the "PTCtIsReaderTest Debugging Project" provided by PIOTEC.

6.1 Command Mode Debugging

6.1.1 Basic Command Mode

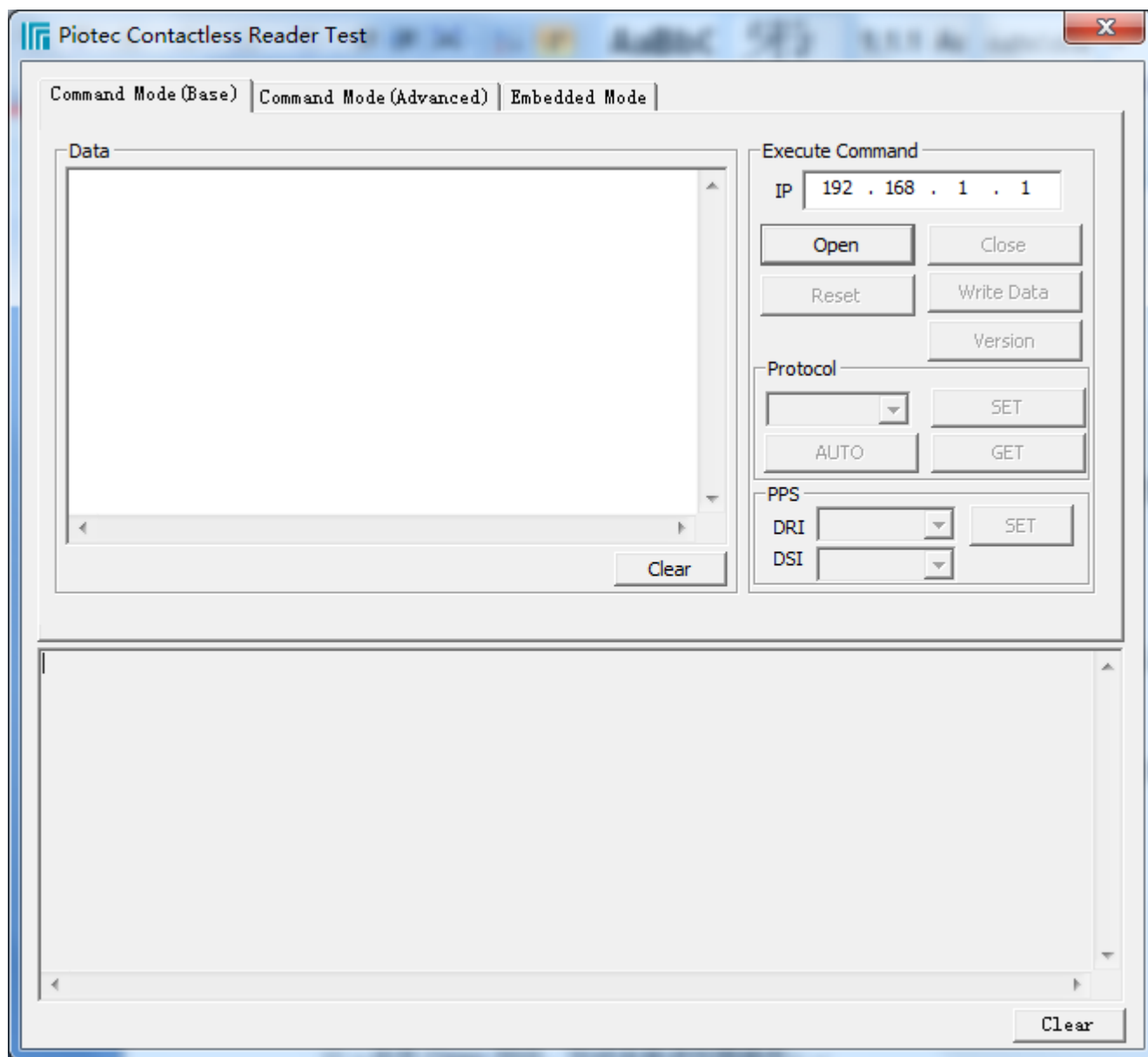


Figure 6.1 Debugging of Basic Command Mode

The card writing test process is as follows:

- 1) Enter the IP address of the reader in the "IP Address Bar".
- 2) Click the "Open" button to complete the operation of connection reader. The 14443A protocol is

used by default.

- 3) Select the communication protocol you want to set in the "Protocol" check box, for example "14443A". Click the "Set" button to complete the settings for the reader communication protocol. (optional)
- 4) The "AUTO" button automatically sets the communication protocol and currently supports 14443A, 14443B and MIFARE cards. (optional)
- 5) Click the "Reset" button to complete the card reset operation.
- 6) Select the "DSI" and "DRI" check boxes in "PPS" to select the communication rate value to be set, for example "106 kBit/s". Click the "Set" button to complete the setting of the reader communication rate. (optional)
- 7) Fill in the APDU command of the write card in the "Data" input box on the left, and click the "Write Data" button to complete the card write operation.
- 8) Click the "Close" button to disconnect the reader.

6.1.2 Advanced Command Mode

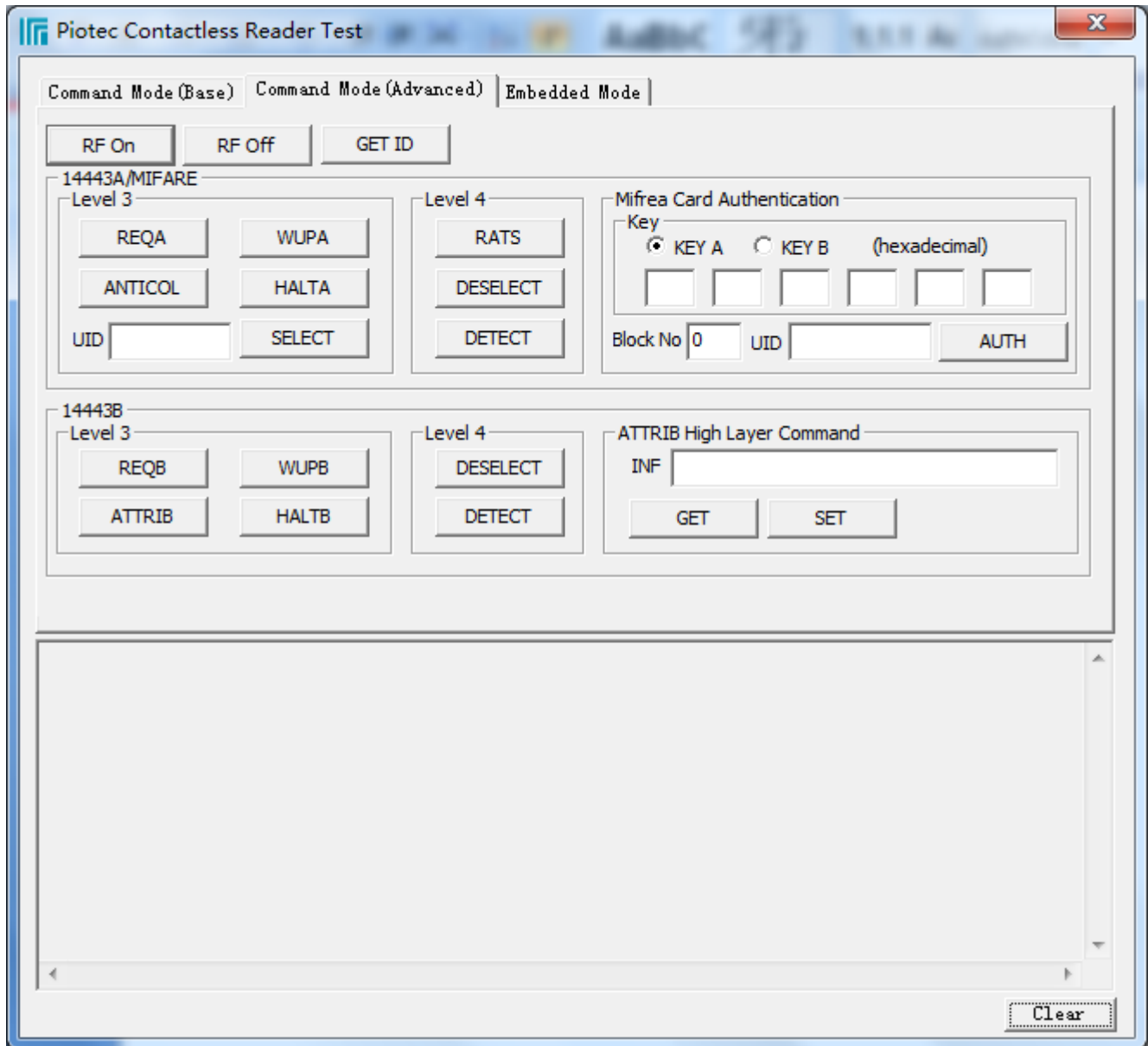


Figure 6.2 Debugging of Advanced Command Mode

14443A test process is as follows:

- 1) In the basic mode interface, enter the IP address of the reader in the "IP address bar" and click the "Open" button to complete the connection reader operation. The 14443A protocol is used by default.
- 2) Click the "RF On" button to turn on the RF carrier operation.
- 3) 点 Click the "REQA" button to complete the request to send a Class A card to the card.

- 4) Click the “HALTA” button to complete the sleep operation of the card. (optional)
- 5) Click the “WUPA” button to complete the wake-up operation of the card. (optional)
- 6) Click the “ANTICOL” button to complete the anti-collision and selection of the card.Or enter the card UID in the "UID edit box" and click the "SELECT" button to complete the selection of the card. Activate the ISO14443-3 layer protocol.
- 7) Click the “RATS” button to complete the request response selection command for the card and activate the ISO14443-4 layer protocol.
- 8) Click the “DETECT” button to see if the card is in the 14443-4 layer protocol. (optional)
- 9) Click the “DESELECT” button to send the S command operation and deactivate the 14443-4 layer protocol. (optional)
- 10) Click the “RF Off” button to turn off the RF carrier operation.
- 11) In the basic mode interface, click the “Close” button to disconnect the reader.

MIFARE test process is as follows:

- 1) In the basic mode interface, Enter the IP address of the reader in the “IP Address Bar” and click the “Open” button to complete the connection of the reader. Select the MIFARE protocol and set it in the "Protocol" checkbox.
- 2) In the basic mode interface, click the "Reset" button to complete the reset operation of the card.
- 3) Select the key type (Class A or Class B) in "Mifare Card Authentication", fill in the secret key (hexadecimal), block number and UID, and click "AUTH" to complete the authentication. As shown in the figure:

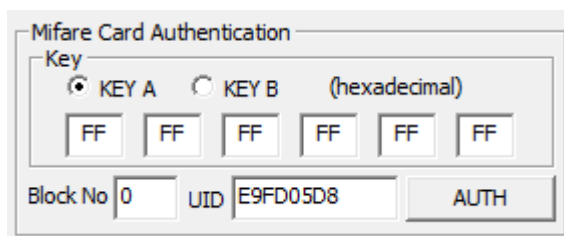


Figure 6.3 MIFARE Authentication of Advanced Command Mode

- 4) In the basic mode interface, click the "Close" button to disconnect the reader.

14443B test process is as follows:

- 1) In the basic mode interface, enter the IP address of the reader in the "IP Address Bar" and click the "Open" button to connect the reader. Select the 14443B protocol and set it in the "Protocol" checkbox.
- 2) Click the "RF On" button to turn on the RF carrier operation.
- 3) Click the "REQB" button to complete the request for sending a Class B card to the card.
- 4) Click the "HALTB" button to complete the sleep operation of the card. (optional)
- 5) Click the "WUPB" button to complete the wake-up operation of the card. (optional)
- 6) Click the "ATTRIB" button to execute the B-type card attrib command to activate the ISO14443-4 layer protocol.
- 7) Click the "DETECT" button to see if the card is in the 14443-4 layer protocol. (optional)
- 8) Click the "DESELECT" button to send the S command operation and deactivate the 14443-4 layer protocol. (optional)
- 9) Click the "RF Off" button to turn off the RF carrier operation.
- 10) In the basic mode interface, click the "Close" button to disconnect the reader.

6.2 Embedded Mode Debugging

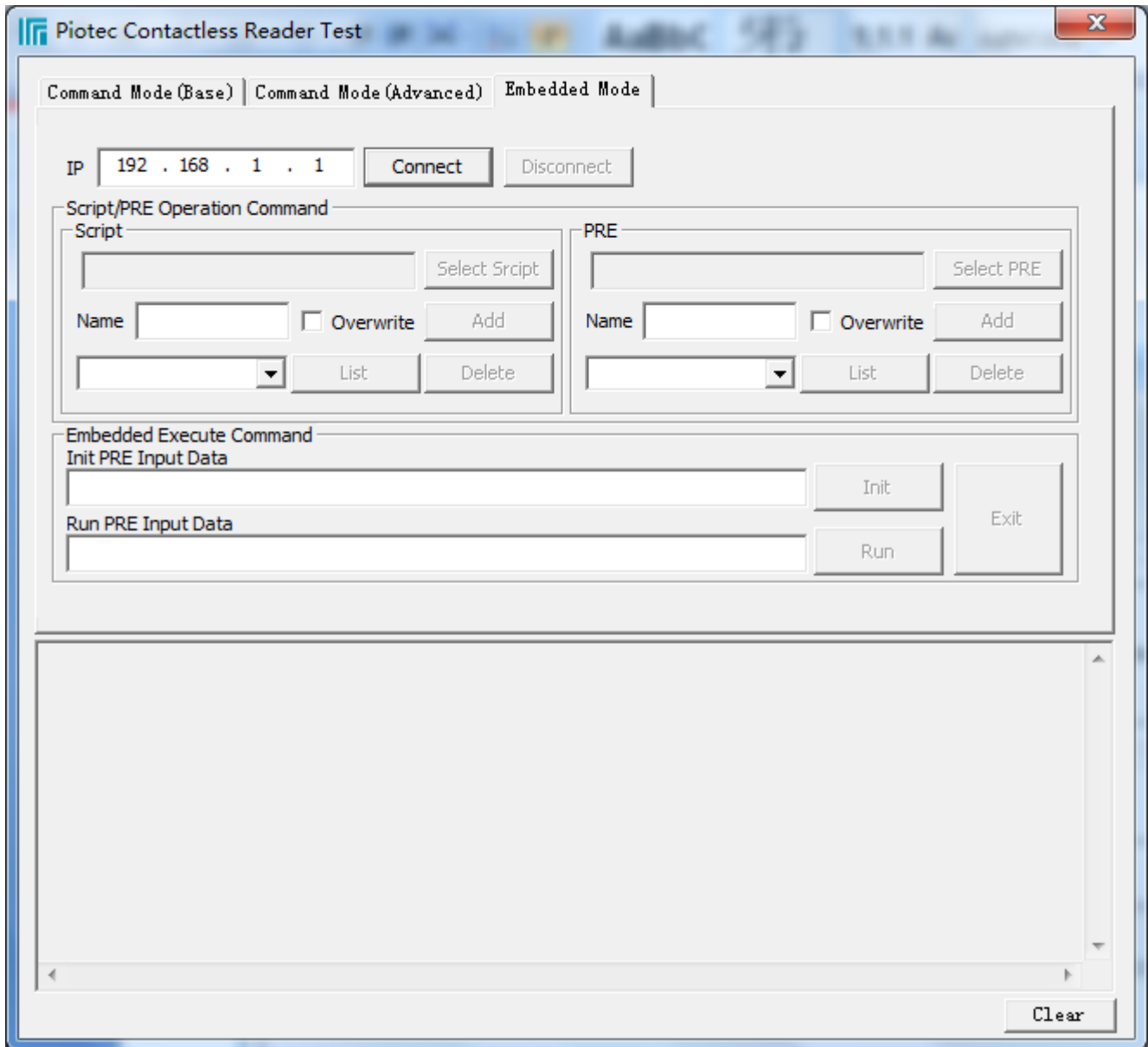


Figure 6.4 Embedded Mode Debugging

Write card test process is as follows:

- 1) Enter the IP address of the reader in the "IP Address Bar".
- 2) Click the "Connect" button to complete the connection reader operation.
- 3) Click the "Select Script" button and the "Select PRE" button to select the script file and PRE file to be downloaded.
- 4) Fill in the name of the script file and PRE file to be downloaded in the "Name edit box", and click

the “Add” button to download the file (Overwrite option indicates whether the duplicate file is overwritten).

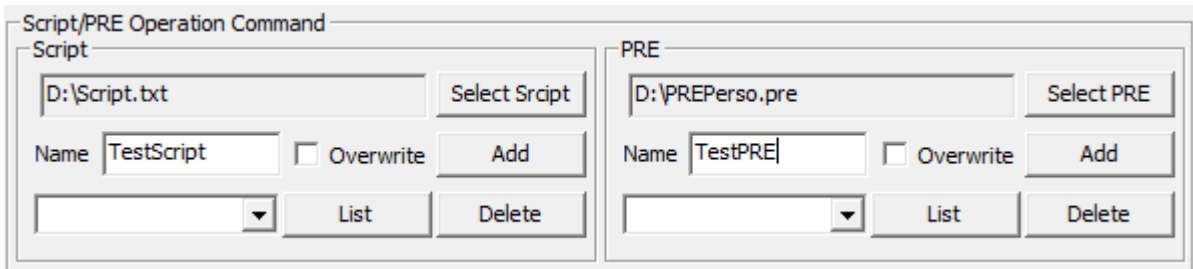


Figure 6.5 Script Operation Interface of Embedded Mode

- 5) Click the "List" button to list the script files and PRE files saved inside the reader. Click the "Delete" button to delete the corresponding file in the check box.



Figure 6.6 Delete Script of Embedded Mode

- 6) Embedded initialization process, select the script file name and PRE file name to be used in the check box, fill in the data that needs to be initialized in the “Init PRE Input Data” edit box (optional), click the “Init” button Perform embedded mode initialization.
- 7) In the embedded running process, fill in the data that needs to be input during runtime in the “Run PRE Input Data” edit box (optional), and click the “Run” button to perform the embedded mode operation.
- 8) Embedded release process, click the “Exit” button to perform the embedded mode release operation.
- 9) Click the “Disconnect” button to disconnect the reader.

Chapter 7 Error Code Explanation



Table 7.1 Error Code Explanation

Error Code	Explanation
0x0000	No error
0x1001	Invalid voltage parameter
0x1002	Power-on interruption
0x1003	ATR timeout
0x1004	Invalid ATR value
0x1005	ATR verification failed
0x1006	Agreement not supported
0x1007	Invalid SLOT
0x1008	Receive data timeout
0x1009	Receive data parity failed
0x1010	Receive LRC failed
0x1011	Receive CRC failed
0x1012	Receive serial number error
0x1013	Failed to set IFSD
0x1014	Hot reset error
0x1015	Illegal status byte
0x1016	Unknown error
0x1017	Block wait timeout

0x1018	Character wait timeout
0x1019	T1 protocol terminates transmission error
0x1020	Exceeded the maximum number of retries
0x1021	Invalid NAD
0x1022	Invalid length
0x1023	Invalid information
0x1024	Invalid working mode
0x1025	Invalid level
0x1026	Unknown control command
0x1027	Does not support PPS exchange
0x1028	SLOT mode not selected
0x1029	Working mode error
0x1030	Power-on mode error
0x1031	Voltage error
0x1032	Card type error
0x1033	Protocol error
0x1034	PPS mode error
0x1035	Receive data overflow
0x1036	Send data timeout
0x1037	Invalid parameter

0x1038	Busy device
0x1039	Failed to get user data
0x1040	Returning user data failed
0x1041	Failed to set frequency
0x1042	Setting WWT failed
0x1043	Setting ignore SW failed
0x1044	Open short circuit test failed
0x2001	Receive data no response timeout
0x2002	CRC or parity error
0x2003	Collision
0x2004	Buffer overflow
0x2005	Invalid frame format
0x2006	Response violation agreement
0x2007	Verification failed
0x2008	Memory read and write error
0x2009	RC temperature overheating
0x2010	RF error
0x2011	RC communication error
0x2012	Wrong length
0x2013	Device resource error

0x2014	Sending side NAK error
0x2015	Receiver NAK error
0x2016	External RF error
0x2017	EMVCoEMD noise error
0x2018	Ignore errors
0x2019	External error
0x2020	Invalid data parameter
0x2021	Invalid parameter
0x2022	Read and write parameter overflow
0x2023	Parameter is not supported
0x2024	Command not supported
0x2025	Condition not supported
0x2026	KEY error
0x2027	Initialization error
0x3001	Device is turned on
0x3002	Failed to open device
0x3003	Device shutdown failed
0x3004	IP error
0x3005	Feature not supported
0x3006	Agreement not supported

0x3007	Parameter error
0x3008	File already exists
0x3009	Failed to open the file
0x3010	Failed to read file
0x3011	Failed to write to file
0x3012	Failed to detect MD5 value
0x3013	Reader model error
0x4001	Unknown communication error
0x4002	IP address is empty
0x4003	Communication connection is not open
0x4004	Communication connection failed to open
0x4005	Communication connection initialization failed
0x4006	Communication connection closed failed
0x4007	Communication connection interruption
0x4008	Communication connection timeout
0x4009	Failed to send data
0x4010	Failed to receive data
0x4011	Data command is not supported
0x4012	Memory error

0x4013	CRC check failed
0x4014	Unknown response
0x4015	Wrong response data length
0x4016	Disconnected from the network
0x4017	Parameter error
0x4018	Communication connection is open

--- End of document ---



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