

SPRINGCARD CONTACTLESS COUPLERS

Developer's reference manual

Headquarters, Europe

SpringCard
13 voie la Cardon
Parc Gutenberg
91120 Palaiseau
FRANCE

Phone : +33 (0) 164 53 20 10
Fax : +33 (0) 164 53 20 18

Americas

SpringCard
6161 El Cajon blvd
Suite B, PMB 437
San Diego, CA 92115
USA

Phone : +1 (713) 261 6746

www.springcard.com

DOCUMENT INFORMATION

Category : Developer's manual
 Group : CSB4/K531/K632/SpringProx
 Reference : PMDE051
 Version : CA
 Status : draft

Keywords :
 CSB4, K531, K632, SpringProx

Abstract :

SpringCard contactless couplers must be driven by an host software by the mean of a serial interface. These manual details the command set supported by the CSB4, K531, K632, SpringProx-CF, and alike.

[pmde051-ca] contactless couplers developer's guide.doc
 saved 15/07/10 - printed 15/07/10

REVISION HISTORY

Ver.	Date	Author	Valid. by Tech.	Qual.	Approv. by	Remarks :
AA	15/03/04	JDA				Early draft
AB	25/05/04	JDA				Some errors corrected
BA	19/06/06	JDA				Added ISO 14443-B and low level interface
BB	22/03/07	JDA				Added listing of error codes
BC	28/02/08	LTC				Added ISO 15693
BD	25/03/09	LTC				Some errors corrected
CA	15/06/10	JDA				New SpringCard layout, major rewriting Added automated card discovery

TABLE OF CONTENT

1.	INTRODUCTION.....	5	9.3.	GET CARD INFORMATION	50
1.1.	ABSTRACT.....	5	10.	STATUS AND ERROR CODES	52
1.2.	IMPORTANT – READ ME FIRST	5	10.1.	SUCCESS AND SPECIAL STATUS	52
1.3.	SUPPORTED PRODUCTS.....	5	10.2.	ERRORS IN RF COMMUNICATION OR PROTOCOL.	52
1.4.	AUDIENCE.....	6	10.3.	ERRORS REPORTED BY THE CARD.....	52
1.5.	SUPPORT AND UPDATES	6	10.4.	ERRORS AT T=CL LEVEL.....	53
1.6.	HARDWARE VERSION WARNING	6	10.5.	OTHER ERRORS.....	53
2.	SERIAL COMMUNICATION.....	7			
2.1.	OVERVIEW	7			
2.2.	THE COMMAND-RESPONSE LAYER	7			
2.3.	THE TRANSPORT LAYER	8			
2.4.	THE PHYSICAL LAYER	12			
3.	BASIC FUNCTION SET	13			
3.1.	DEVICE INFORMATION AND CONTROL.....	13			
3.2.	RF INTERFACE CONFIGURATION AND CONTROL ..	16			
3.3.	NON-VOLATILE CONFIGURATION (EEPROM)	18			
3.4.	LEDs, BUZZER AND I/Os FUNCTIONS	19			
4.	ISO 14443-A AND MIFARE.....	22			
4.1.	ISO 14443-A CARD CONTROL	22			
4.2.	ISO 14443-A FRAME EXCHANGE	24			
4.3.	MIFARE CLASSIC OPERATION, KEYS STORED IN THE COUPLER	25			
4.4.	MIFARE CLASSIC OPERATION, KEYS PROVIDED BY THE HOST	29			
4.5.	MIFARE ULTRALIGHT OPERATION	31			
5.	ISO 14443-B.....	32			
5.1.	ISO 14443-B CARD CONTROL	32			
5.2.	ISO 14443-B FRAME EXCHANGE	34			
6.	“T=CL” (ISO 14443-4)	35			
6.1.	T=CL ACTIVATION – ISO 14443-A	35			
6.2.	T=CL ACTIVATION – ISO 14443-B	37			
6.3.	T=CL APDU EXCHANGE.....	38			
6.4.	T=CL DESELECT	38			
7.	ISO 15693.....	39			
7.1.	ISO 15693 CARD CONTROL	39			
7.2.	ISO 15693 FRAME EXCHANGE	41			
7.3.	ISO 15693 READ.....	42			
7.4.	ISO 15693 WRITE AND LOCK	43			
8.	OTHER RF PROTOCOLS	44			
8.1.	INNOVATRON RADIO PROTOCOL (CALYPSO)	44			
8.2.	NXP ICODE1	44			
9.	AUTOMATIC CARD DISCOVERY	45			
9.1.	FIND CARD	45			
9.2.	POLLING LOOPS	47			

1. INTRODUCTION

1.1. ABSTRACT

This document provides all necessary information to operate a SpringCard contactless coupler through its serial interface (communication protocols, function sets).

1.2. IMPORTANT – READ ME FIRST

SpringCard has developed a comprehensive and complete software library that implements everything that is described in this document (communication protocols, host-side high-level API), and more¹.

This software library is available free of charge in the relevant SDKs, both as binary (springprox.dll for Windows) and as source code (ANSI C).

Using SpringProx API is the recommended solution to operate with SpringCard contactless couplers. It facilitates integration and limits software development effort, thanks to the samples supplied in the SDK that could be used immediately.

So, please think twice before developing from scratch brand new software to operate our contactless couplers. Keep this document as a reference, but prefer whenever it is possible high-level function calls and avoid direct access to the coupler.

1.3. SUPPORTED PRODUCTS

At the date of writing, this document refers to every SpringCard contactless product running a "legacy" firmware **version ≥ 1.51 ²** :

- SpringProx-CF, SpringProx-RC, etc,
- K531 and K632,
- CSB4,
- CSB6 family in Legacy mode (including CrazyWriter, Prox'N'Roll, SpringWAP, etc).

Please refer to the product leaflet and the integration guide of each product, for accurate specification and a detailed list of features.

¹ An important feature of the SpringProx API is its "knowledge" of the various software and hardware versions, and its ability to overcome silently (and most of the time efficiently) a function that is missing or that is known to have limitations in some versions, as the present document only covers the version that is currently shipping at the date of writing.

² For older products: either upgrade the firmware with an up-to-date version, or refer to earlier versions of this document.

1.4. AUDIENCE

This manual is designed for use by application developers. It assumes that the reader has expert knowledge of computer development.

1.5. SUPPORT AND UPDATES

Interesting related materials (product datasheets, application notes, sample software, HOWTOs and FAQs...) are available at SpringCard's web site:

www.springcard.com

Updated versions of this document and others will be posted on this web site as soon as they are made available.

For technical support enquiries, please refer to SpringCard support page, on the web at address www.springcard.com/support .

1.6. HARDWARE VERSION WARNING

Note that SpringCard contactless products could be divided in two different hardware groups:

- Hardware built over the NXP RC531 chipset support only contactless cards in the ISO 14443 family (Mifare, Desfire, ICAO passports and others travel documents, Calypso or equivalent cards for public transport, payment cards...),
- Hardware built over the RC632 chipset also support vicinity cards and RFID tags in the ISO 15693 family (ICODE, TagIT, ...).

This documentation covers both groups, but of course the ISO 15693-related functions are not available on products based on the RC531 chipset.

NB: the actual features supported by a coupler are exposed to the host thanks to the Get Coupler Capabilities command (see 3.1.2 and Table 1).

2. SERIAL COMMUNICATION

2.1. OVERVIEW

The SpringProx contactless couplers follow a master-slave scheme, where the coupler is the slave of a 'host' (either a computer or a microcontroller).

Therefore, the communication flow is driven by the host: a Command from the host is followed (after necessary processing time) by a Response from the coupler.

The communications follow a three-layer scheme:

- The Command-Response layer,
- The Transport layer,
- The Physical layer.

2.2. THE COMMAND-RESPONSE LAYER

The Command-Response layer handles and interprets the SpringProx commands and responses.

Chapters 3 to 10 fully describe the Command set and the Response format.

a. Format

Commands and Response are made of three fields:

- The **command code** (CMD) for a Command or the **status code** (STA) for a Response,
- The size of the data (LEN),
- The **data** itself, if some (DATA).

b. Status codes

The status returned by the coupler in its Response is explained on a single byte, from h_{00} to h_{80} .

For coherence with the documentation of the SpringProx API, the status codes are **documented as negative decimal values**.

This is only a convention. Translation from one representation to another is trivial:

- $h_{00} \Leftrightarrow 0$ means "Success",
- h_{01} to $h_{7F} \Leftrightarrow -1$ to -127 are warning and error codes (see chapter 10 for the complete list),
- $h_{80} \Leftrightarrow -128$ means "time extension" (see paragraph d below).

c. Length

When size of data is lower than 128, then the LEN field is expressed as a single byte (*value between 0 and 127*).

When size of data is between 128 and 255, the LEN field is expressed as 2 bytes:

- The 1st byte is fixed to $\text{h}80$ (128)
- The 2nd byte is equal to LEN minus $\text{h}80$ (*value between 0 and 127*).

When size of data is 256 or greater, the LEN field is expressed as 3 bytes:

- The 1st byte is fixed to $\text{h}80$ (128)
- The 2nd byte is fixed to $\text{h}80$ (128)
- The 3rd byte is equal to LEN minus $\text{h}100$ (*value between 0 and 127*).

d. Processing timeout and Time Extension

The coupler shall always send its Response within 1000ms after receiving the Command³.

If the processing is not terminated within 1000ms, the coupler sends a Time Extension frame. The Time Extension frame is repeated every 600ms to 800ms until processing is terminated; then the actual Response takes place.

The host shall not answer the Time Extension frame, nor send any new Command while the coupler is still processing the previous Command.

2.3. THE TRANSPORT LAYER

The Transport layer handles message addressing, specifies the transmission type, and validates every transmission. The transport layer can use one out of four protocols.

- The ASCII protocol,
- The (modified) OSI3964R protocol,
- The "fast" protocol,
- The "bus" protocol.

The "fast" protocol is the recommended choice.

NB: the "bus" protocol is only available in the firmware of products based on an RS-485 physical interface (see *Table 1: Coupler's capabilities*).

a. Protocol selection

The choice of the protocol is up to the host. Upon start-up, the coupler is ready to accept all protocols, and will answer using the same protocol as used by the host.

³ The returning Wait One Card (§ 9.2.1) and Wait Multiple Cards (9.2.2) are the two only exception to this rule.

b. Protocol change

The host may switch at any time from ASCII protocol to either OSI or "Fast" protocols. Reverting to ASCII protocol from another is not possible (until the coupler resets).

The host may switch at any time from OSI protocol to "Fast" protocol. Reverting to OSI protocol from "Fast" is not possible (until the coupler resets).

2.3.2. The ASCII protocol

This is a lightweight protocol, easy to implement in a host with small RAM capacity, or even manually under a terminal-emulation program (HyperTerminal for instance).

The transmission is made in ASCII Hexadecimal ($_{h12}$ $_{h34}$ $_{h56}$ is transmitted as string "123456"). The only valid characters are '0' to '9', 'A' to 'F' or 'a' to 'f', and the characters used for control ('\$', '+', '-', {CR}, {LF}). Any other character would be discarded.

a. Host to coupler

\$ <CMD_{ASC}> <LEN_{ASC}> [<DATA_{ASC}>] {CR} [{LF}]

The frame is prefixed by the ASCII dollar sign ('\$') and terminated by a Carriage Return (CR). The Line Feed (LF) is optional.

The coupler immediately acknowledges the frame with an ASCII plus sign ('+').

If the coupler detects a communication error, it N-acknowledges with an ASCII minus sign ('-') followed by a NACK-code (see § 2.3.6).

b. Coupler to host

<abs(STA)_{ASC}> <LEN_{ASC}> [<DATA_{ASC}>] {CR} {LF}

c. Time Extension

The Time Extension frame is an ASCII plus sign ('+').

d. Communication timeouts

There's no communication timeout associated to this protocol.

2.3.3. The (modified) OSI3964R protocol

The OSI 3964R protocol is a feature-rich network protocol. The implementation in the coupler has strong limitations due to memory and code-size constraints.

NB: Using this protocol is not recommended in new projects.

a. Principles

At first, the sender sends the ASCII {STX} character ($_{h}02$). The receiver confirms it is present and listening, sending ASCII {DLE} character ($_{h}10$).

The payload starts by a Sequence number (SEQ), followed by the Command or Status (CMD or STA), then LEN and DATA. It is ended by a LRC, and terminated by the ASCII {DLE} {ETX} sequence ($\{ETX\} = _{h}03$).

The receiver acknowledges the frame by sending {DLE} again. When the receiver detects a communication error, instead of sending {DLE} it sends {NAK} ($_{h}15$) followed by an error code (see § 2.3.6).

In the payload itself (from <SEQ> to <LRC> included), every byte equal to one of the protocol-reserved values ({DLE}, {STX} and {ETX}) shall be prefixed by a {DLE} protocol byte.

b. Host to coupler

<i>Host</i>	<i>Coupler</i>
{STX}	
	{DLE}
<SEQ> <CMD> <LEN> [<DATA>] <LRC> {DLE} {ETX}	
	{DLE}

Where <LRC> = XOR (<SEQ> <CMD> <LEN> [<DATA>])

The host shall increment its Sequence number after each exchange.

c. Coupler to host

<i>Coupler</i>	<i>Host</i>
{STX}	
	{DLE}
<SEQ> <CMD> <LEN> [<DATA>] <LRC> {DLE} {ETX}	
	{DLE}

Where <LRC> = XOR (<SEQ> <CMD> <LEN> [<DATA>])

In its Response, the coupler always echoes the Sequence number received within the Command.

d. Time Extension

The Time Extension frame is a single ASCII {DLE} byte. The host shall not acknowledge this byte.

e. Communication timeouts

- Sender {STX} to receiver {DLE} : 20ms,
- Sender {DLE} {ETX} to receiver {DLE} : 20ms,
- Inter-byte timeout: 5ms.

2.3.4. The "fast" protocol

This protocol is designed for high-speed communication on a reliable physical layer. It provides frame synchronization and checking but no software flow control.

Every frame starts with the ASCII {SYN} character ($_{h}16$), followed by a Sequence number (SEQ). The host shall increment its Sequence number after each exchange. In its Response, the coupler always echoes the Sequence number received within the Command.

The LRC field helps detecting communication errors.

a. Host to coupler

{SYN} <SEQ> <CMD> <LEN> [<DATA>] <LRC>

Where <LRC> = XOR (<SEQ> <CMD> <LEN> [<DATA>])

When the coupler detects a communication error, it sends {NAK} ($_{h}15$) followed by an error code (see § 2.3.6). When no error is detected, the frame is not acknowledged.

b. Coupler to host

{SYN} <SEQ> <STA> <LEN> [<DATA>] <LRC>

Where <LRC> = XOR (<SEQ> <CMD> <LEN> [<DATA>])

If the host detects a communication error, it may either send the same Command again, or ask the coupler to repeat its Response by sending a Repeat Command (CMD= $_{h}80$ and LEN= $_{h}00$) with the same Sequence number. When no error is detected, the frame is not acknowledged.

c. Time Extension

The Time Extension frame is formatted as a standard Response frame, having STA= $_{h}80$ and LEN= $_{h}00$ (and the current Sequence number).

d. Communication timeouts

The transmission timeout (total frame duration from {SYN} to <LRC>) is fixed to 400ms.

There's no inter-byte timeout associated to this protocol.

2.3.5. The "Bus" protocol

To be written

2.3.6. Transport layer error codes

NACK code	Meaning
h09	Overrun: at least one byte has been lost
h0A	Length error: the length of the frame doesn't match the LEN value
h0B	LRC error: computed LRC doesn't match the LRC value
h0C	Buffer overflow: frame is longer than supported
h0D	Protocol error: an invalid value has been received
h0E	Timeout error: a communication timeout has expired while receiving
h0F	Hardware error: fatal UART error (check physical line)

2.4. THE PHYSICAL LAYER

The Physical layer handles the data transmission itself. The physical layer uses an asynchronous serial protocol. The actual implementation depends on the hardware associated to the K531/K632 module (RS-232, RS at TTL level, RS over USB, RS-485, ...).

The default configuration is 38400bps, 8 data bits, 1 stop bit, no parity, and no flow control.

3. BASIC FUNCTION SET

3.1. DEVICE INFORMATION AND CONTROL

3.1.1. Get Firmware Information

Command

CMD	LEN
h4F	h00

Response OK

STA	LEN	0..3	4	5	6	7..11	12..15
0	h10	<i>Product ID</i>	<i>Ver. MSB</i>	<i>Ver. LSB</i>	<i>Build no.</i>	<i>Chipset info</i>	<i>Serial number</i>

- Product ID: either "CSB4", "K531", "K632", "SPX2", etc,
- Version is expressed as three bytes: *MSB.LSB [Build No]* ,
- *Chipset info* is the data identifier of the NXP RC531 or RC632 chipset,
- *Serial number* is the serial number of the NXP RC531 or RC632 chipset. This is used as the serial number of the product itself.

3.1.2. Get Coupler Capabilities

Command

CMD	LEN
h50	h00

Response OK

STA	LEN	0..3
0	h04	<i>Capabilities</i>

Capabilities is a 32-bit field (MSB-first). Bits that are set denote the actual features offered by the coupler. Refer to Table 1 on next page.

Table 1: Coupler's capabilities

BIT	MASK	Meaning
0	h00000001	<i>The coupler supports ISO 14443⁴</i>
1	h00000002	<i>The coupler supports ISO 15693⁵</i>
2	h00000004	<i>RFU</i>
3	h00000008	<i>RFU</i>
4	h00000010	<i>RFU</i>
5	h00000020	<i>The coupler supports ISO 7816 (smartcard interface)⁶</i>
6	h00000040	<i>Deprecated (CSB5-Bio / SpringProx-RC-Bio)</i>
7	h00000080	<i>RFU</i>
8	h00000100	<i>The coupler implements the ASCII protocol (§)</i>
9	h00000200	<i>The coupler implements the "Fast" protocol (§)</i>
10	h00000400	<i>RFU</i>
11	h00000800	<i>The coupler implements the "Bus" protocol (§)</i>
12	h00001000	<i>The coupler's USER pin drives a RS-485 line buffer</i>
13	h00002000	<i>The coupler supports the Repeat command</i>
14	h00004000	<i>The coupler supports 115200bps baudrate</i>
15	h00008000	<i>Deprecated (early CSB4-S)</i>
16	h00010000	<i>The coupler has an USB interface / CDC-ACM profile</i>
17	h00020000	<i>The coupler has an USB interface / CCID profile (PC/SC)</i>
18	h00040000	<i>The coupler has an USB interface / HID profile (keyboard)</i>
19	h00080000	<i>RFU</i>
20	h00100000	<i>The coupler has an EEPROM to store its configuration</i>
21	h00200000	<i>RFU</i>
22	h00400000	<i>RFU</i>
23	h00800000	<i>RFU</i>
24	h01000000	<i>The coupler features a "console"</i>
25	h02000000	<i>Deprecated (early CSB4-S)</i>
26	h04000000	<i>RFU</i>
27	h08000000	<i>RFU</i>
28	h10000000	<i>RFU</i>
29	h20000000	<i>RFU</i>
30	h40000000	<i>RFU</i>
31	h80000000	<i>RFU</i>

⁴ This bit is always set for contactless couplers!

⁵ This bit is set only if the coupler is based on NXP RC632 chipset

⁶ This bit is set for CSB5, SpringProx-RC, SpringWAP, CSB6, CrazyWriter, EasyFinger

3.1.3. Reset Coupler

Command

CMD	LEN	0	1	2	3
h9F	h04	hDE	hAD	hDE	hAD

NB: the coupler resets immediately, so it never answers this Command. Upon start-up, the coupler sends its *Product ID* at 38400bps and waits for the first Command. ISO 14443-A mode is selected.

3.1.4. Change Baudrate

NB: the 115200bps baudrate is not supported by all couplers. Before trying to choose this baudrate, use the Get Coupler Capabilities command to verify that the connected coupler supports it (see 3.1.2 and Table 1).

3.2. RF INTERFACE CONFIGURATION AND CONTROL

3.2.1. Select RF Mode

Command

CMD	LEN	0	1
h58	h02	h0C	<i>Mode</i>

Mode could take one of the following values:

- h01 : ISO 14443-A,
- h02 : ISO 14443-B,
- h03 : Innovatron radio protocol (Calypso),
- h04 : ISO 15693, fast,
- h05 : ICODE1, fast,
- h14 : ISO 15693, slow,
- h15 : ICODE1, slow.

NB: only the products based on NXP RC632 do support the ISO 15693 and ICODE1 modes. Use the Get Coupler Capabilities command to check whether or not the connected coupler supports it (see 3.1.2 and Table 1).

Response OK

STA	LEN
0	h00

3.2.2. RF Field ON/OFF

Command

CMD	LEN	0	0
h58	h02	h0A	<i>Flag</i>

Flag could take one of the following values:

- h00 : RF field is switched OFF,
- h01 : RF field is switched ON.

Response OK

STA	LEN
0	h00

3.2.3. Read RC Register

Command

CMD	LEN	0
hAB	h01	<i>Addr</i>

Response OK

STA	LEN	0
0	h01	<i>Value</i>

3.2.4. Write RC Register

Command

CMD	LEN	0	1
hAB	h02	<i>Addr</i>	<i>Value</i>

Response OK

STA	LEN
0	h00

3.2.5. Reset RF Interface

Command

CMD	LEN
h9F	h00

Response OK

STA	LEN
0	h00

NB: ISO 14443-A mode is selected when the RF interface resets.

3.3. NON-VOLATILE CONFIGURATION (EEPROM)

NB: not all couplers have an EEPROM to store their non-volatile configuration. Before invoking the *Read Configuration Register / Write Configuration Register* functions, check that the connected coupler actually supports them, using the *Get Coupler Capabilities* command (3.1.2 and Table 1).

3.3.1. Read Configuration Register

Command

CMD	LEN	0	1
h58	h02	h0E	Addr

Response OK

STA	LEN	0..xx-1
0	xx	Value

3.3.2. Write Configuration Register

Command

CMD	LEN	0	1	2..xx-1
h58	xx	h0D	Addr	Value

Response OK

STA	LEN
0	h00

3.4. LEDs, BUZZER AND I/Os FUNCTIONS

3.4.1. Set LEDs

Command coupler with 2 LEDs

CMD	LEN	0	1	2
h58	h03	h1E	Red	Green

Command coupler with 3 LEDs

CMD	LEN	0	1	2	3
h58	h04	h1E	Red	Green	Yellow / blue

Every LED control byte (*Red*, *Green* or *Yellow/blue*) could take one of the following values:

- h00 : switched OFF,
- h01 : switched ON,
- h02 : slow blinking,
- h04 : fast blinking,
- h05 : "heart beat",
- h06 : slow blinking, inverted,
- h07 : fast blinking, inverted,
- h08 : "heart beat", inverted,
- h09 : switched ON, half intensity.

Response OK

STA	LEN
0	h00

3.4.2. Set Buzzer

Command

CMD	LEN	0	1..2
h58	h03	h1C	Duration (ms)

Response OK

STA	LEN
0	h00

3.4.3. Set USER

This function configures the USER pin as output and defines its state.

NB: the USER pin is not available on all hardware. Before invoking this function, check in device documentation whether it is available or not.

Command

CMD	LEN	0	1
h58	h02	h1F	<i>Flag</i>

Flag could take one of the following values:

- h00 : set USER pin at LOW level,
- h01 : set USER pin at HIGH level.

Response OK

STA	LEN
0	h00

3.4.4. Get USER

This function configures the USER pin as input and returns its current state.

NB: the USER pin is not available on all hardware. Before invoking this function, check in device documentation whether it is available or not.

Command

CMD	LEN	0
h58	h01	h1F

Response OK

STA	LEN	0
0	h01	<i>Flag</i>

Flag takes one of the following values:

- h00 : USER pin is at LOW level,
- h01 : USER pin is at HIGH level.

3.4.5. Get MODE

This function configures the MODE pin as input and returns its current state.

NB: the MODE pin is not available on all hardware. Before invoking this function, check in device documentation whether it is available or not.

Command

CMD	LEN	O
h58	h01	h1D

Response OK

STA	LEN	O
0	h01	Flag

Flag takes one of the following values:

- h00 : MODE pin is at LOW level,
- h01 : MODE pin is at HIGH level.

4. ISO 14443-A AND MIFARE

The functions listed in this chapter are available when the coupler's RF interface has been configured for ISO 14443-A mode (see § 3.2.1).

Note that this is the default mode upon start-up.

4.1. ISO 14443-A CARD CONTROL

Please refer to ISO 14443-3 standard (type A) for details on this chapter.

4.1.1. Activate Idle (REQA / Anticoll / Select)

This function performs the standard activation loop. Only cards in the IDLE state could be activated.

Thanks to the anti-collision scheme, only one card will be selected, even if there are numerous cards in front of the antenna.

Command

CMD	LEN
h4D	h00

Response *no card in the field*

STA	LEN
-1	h00

Response *card found, 4-byte UID*

STA	LEN	0..3	4..6	5
0	h06	UID	ATQ	SAK

Response *card found, 7-byte UID*

STA	LEN	0..6	7..8	9
0	h0A	UID	ATQ	SAK

Response *card found, 10-byte UID*

STA	LEN	0..9	10..11	12
0	h0D	UID	ATQ	SAK

4.1.2. Activate Any (WUPA / Anticoll / Select)

This function performs the standard ISO 14443-A level 3 activation loop. Cards either in the IDLE or the HALTED state could be activated.

Thanks to the anti-collision scheme, only one card will be selected, even if there are numerous cards in front of the antenna.

Command

CMD	LEN
h40	h00

Responses: same as 4.1.1

4.1.3. Activate Again (WUPA / Select)

This function tries to wake-up and select again a card whose UID is already known.

Command *activate last card again*

CMD	LEN
h44	h00

Command *activate a specific card again*

CMD	LEN	0..xx-1
h44	xx	UID

xx (size of UID) could be either h04, h07 or h0A.

Response *OK*

STA	LEN
0	h00

Response *no answer*

STA	LEN
-1	h00

4.1.4. Halt

This command puts the currently activated card into the HALT state.

Command

CMD	LEN
h45	h00

Response OK

STA	LEN
0	h00

4.2. ISO 14443-A FRAME EXCHANGE

4.2.1. Standard Frame Exchange

Command

CMD	LEN	0..1	2..3	4	5..6	6..xx-1
h94	hXX	<i>Send Len</i>	<i>Recv Len</i>	h01	<i>Timeout</i>	<i>Card's command</i>

Response OK

STA	LEN	0..1	2..xx-1
0	hXX	<i>Recv Len</i>	<i>Card's Response</i>

4.2.2. Advanced Frame Exchange

To be written

4.3. MIFARE CLASSIC OPERATION, KEYS STORED IN THE COUPLER

The functions listed here are related to NXP Mifare Classic cards (1K, 4K, Mini, Mifare Plus running in Level 1).

A good understanding of Mifare Classic memory mapping and authentication scheme is required to work with those cards. Please refer to relevant NXP's documentations for details on function parameters and usages.

4.3.1. Load Key in Secure EEPROM

The coupler is able to store permanently 16 'A' keys and 16 'B' keys in a secure EEPROM (inside the NXP RC531 or RC632 chipset) to get automatically authenticated onto Mifare cards.

Command

CMD	LEN	0	1	2..7
hA8	h08	Key type	Key index	Key value

Key type could take one of the following values:

- h00 : type 'A' key,
- h01 : type 'B' key.

Key index could take one of the following values:

- h00 : 1st key,
- h01 : 2nd key,
- ...
- h0F : 16th key.

(A total of 32 keys -16 'A' and 16 'B'- could be stored).

Response OK

STA	LEN
0	h00

4.3.2. Load Key in RAM

The coupler is able to store 4 'A' keys and 4 'B' keys in its RAM to get automatically authenticated onto Mifare cards. Those keys are lost upon reset.

Command

CMD	LEN	0	1..7	8..13
$h4C$	$h0E$	<i>Key ID</i>	<i>(dummy)</i>	<i>Key value</i>

Key ID could take one of the following values:

- $h00$: 1st key A in RAM,
- $h01$: 2nd key A in RAM,
- $h02$: 3rd key A in RAM,
- $h03$: 4th key A in RAM,
- $h04$: 1st key B in RAM,
- $h05$: 2nd key B in RAM,
- $h06$: 3rd key B in RAM,
- $h07$: 4th key A in RAM.

Response OK

STA	LEN
$h00$	$h00$

4.3.3. Read Block

This function reads one block. All Mifare keys known by the readers (RAM and EEPROM) are tried in sequence until an authentication succeeds.

NB: depending on the location of the correct key in the list, this could take a long time (up to 1s per sector). Whenever this is possible, let the host specify the key (see 4.4.1).

Command

CMD	LEN	0
$h49$	$h01$	<i>Block</i>

Response OK

STA	LEN	0..15
0	$h10$	<i>Block data</i>

4.3.4. Write Block

This function writes one block. All Mifare keys known by the readers (RAM and EEPROM) are tried in sequence until an authentication succeeds.

NB: depending on the location of the correct key in the list, this could take a long time (up to 1s per sector). Whenever this is possible, let the host specify the key (see 4.4.2).

This function makes it possible to write sector's trailer (special blocks holding the access conditions and the authentication keys of the sector).

Always format block's content according to Mifare documentation when working with such blocks. Be aware that writing invalid data in a sector's trailer is likely to make the sector permanently unusable.

Command

CMD	LEN	0	1..16
h4B	h11	Block	Block data

Response OK

STA	LEN
0	h00

4.3.5. Read Sector

This function reads all the blocks of the specified sector (but sector's trailer). All Mifare keys known by the readers (RAM and EEPROM) are tried in sequence until an authentication succeeded.

NB: depending on the place of the correct key in the list, this could take a long time (up to 1s per sector). Whenever this is possible, let the host specify the key (see 4.4.3).

Command

CMD	LEN	0
h48	h01	Sector

Response OK, 3-block sector

STA	LEN	0..47
0	h30	Sector data

Response OK, 15-block sector

STA	LEN	0..239
0	hF0	Sector data

4.3.6. Write Sector

This function writes one sector. All Mifare keys known by the readers (RAM and EEPROM) are tried in sequence until an authentication succeeds.

NB: depending on the location of the correct key in the list, this could take a long time (up to 1s per sector). Whenever this is possible, let the host specify the key (see 4.4.2).

Command 3-block sector

CMD	LEN	O	1..48
h4A	h31	Sector	Sector data

Command 15-block sector

CMD	LEN	O	1..240
h4A	hF1	Sector	Sector data

Response OK

STA	LEN
0	h00

4.3.7. Get Authentication Information

Command

CMD	LEN	O
h58	h01	h33

Response OK

STA	LEN	O
0	h01	Info

4.4. MIFARE CLASSIC OPERATION, KEYS PROVIDED BY THE HOST

The functions listed here are related to NXP Mifare Classic cards (1K, 4K, Mini, Mifare Plus running in Level 1).

A good understanding of Mifare Classic memory mapping and authentication scheme is required to work with those cards. Please refer to relevant NXP's documentations for details on function parameters and usages.

4.4.1. Read Block

This function reads one block, using the key provided by the host.

NB: the supplied key is first tried as a type 'A' key, and as a type 'B' key only upon authentication error. Therefore, reading a block using its 'B' key takes longer than reading the same block using its 'A' key.

Command

CMD	LEN	0	1..6
h49	h07	Block	Key value

Response OK

STA	LEN	0..15
0	h10	Block data

4.4.2. Write Block

This function writes one block, using the key provided by the host.

NB: the supplied key is first tried as a type 'B' key, and as a type 'A' key only upon authentication error. Therefore, writing a block using its 'A' key takes longer than writing the same block using its 'B' key.

This function makes it possible to write sector's trailer (special blocks holding the access conditions and the authentication keys of the sector).

Always format block's content according to Mifare documentation when working with such blocks. Be aware that writing invalid data in a sector's trailer is likely to make the sector permanently unusable.

Command

CMD	LEN	0	1..16	17..22
h4B	h17	Block	Block data	Key value

Response OK

STA	LEN
0	h00

4.4.3. Read Sector

This function reads one sector, using the key provided by the host.

NB: the supplied key is first tried as a type 'A' key, and as a type 'B' key only upon authentication error. Therefore, reading a sector using its 'B' key takes longer than reading the same sector using its 'A' key.

Command

CMD	LEN	0	1..6
$h48$	$h07$	Sector	Key value

Response OK, 3-block sector

STA	LEN	0..47
0	$h30$	Sector data

Response OK, 15-block sector

STA	LEN	0..239
0	$hF0$	Sector data

4.4.4. Write Sector

This function writes one sector, using the key provided by the host.

NB: the supplied key is first tried as a type 'B' key, and as a type 'A' key only upon authentication error. Therefore, writing a block using its 'A' key takes longer than writing the same block using its 'B' key.

Command 3-block sector

CMD	LEN	0	1..48	49..54
$h4A$	$h37$	Sector	Sector data	Key value

Command 15-block sector

CMD	LEN	0	1..240	241..246
$h4A$	$hF7$	Sector	Sector data	Key value

Response OK

STA	LEN
0	$h00$

4.5. MIFARE ULTRALIGHT OPERATION

The functions listed here are related to NXP Mifare UltraLight cards (and UltraLight C).

Please refer to relevant NXP's documentations for details on function parameters and usages.

4.5.1. Read 4 Pages

Command

CMD	LEN	0
h46	h01	1 st Page

Response OK

STA	LEN	0..15
0	h10	Data

4.5.2. Write Page

Command

CMD	LEN	0	1..4
h47	h05	Page	Data

Response OK

STA	LEN
0	h00

5. ISO 14443-B

The functions listed in this chapter are available when the coupler's RF interface has been configured for ISO 14443-B mode (see § 3.2.1).

5.1. ISO 14443-B CARD CONTROL

Please refer to ISO 14443-3 standard (type B) for details on this chapter.

5.1.1. Activate Idle (REQB)

This function implements REQB. Only cards in the IDLE state could be activated. There's no anti-collision in this mode.

Command

CMD	LEN
h4D	h00

Response *no card in the field*

STA	LEN
-1	h00

Response *card found*

STA	LEN	0..10
0	h0B	ATQ

NB: the four first bytes of the ATQ are the card's PUPI.

5.1.2. Activate Any (WUPB)

This function implements WUPB. Cards either in the IDLE or the HALTED state could be activated. There's no anti-collision in this mode.

Command

CMD	LEN
h4D	h00

Responses: same as 5.1.1

5.1.3. Activate Again (WUPB / Select)

This function tries to wake-up and select again a card whose UID is already known.

Command *activate last card again*

CMD	LEN
h44	h00

Response *OK*

STA	LEN
0	h00

Response *no answer*

STA	LEN
-1	h00

5.1.4. Halt

This command puts the currently activated card into the HALT state.

Command

CMD	LEN
h45	h00

Response *OK*

STA	LEN
0	h00

5.2. ISO 14443-B FRAME EXCHANGE

5.2.1. Standard Frame Exchange

Command

CMD	LEN	0..1	2..3	4	5..6	6..xx-1
<i>h97</i>	<i>hXX</i>	<i>Send Len</i>	<i>Recv Len</i>	<i>h01</i>	<i>Timeout</i>	<i>Card's command</i>

Response OK

STA	LEN	0..1	2..xx-1
<i>0</i>	<i>hXX</i>	<i>Recv Len</i>	<i>Card's Response</i>

5.2.2. Advanced Frame Exchange

To be written

6. "T=CL" (ISO 14443-4)

ISO 14443 level 4 (a.k.a. "T=CL") is a high-level protocol to exchange application-level buffers (APDU) between the host and the card.

As the coupler fully implements this protocol in its firmware, the complexity of the protocol is totally masked to the host.

ISO 14443 level 4 allows a single coupler to communicate with more than one card at once (alternatively), thanks to an address field called CID (Card Identifier). The coupler is able to manage up to 14 active CIDs at the same time.

Please refer to ISO 14443-4 standard for details on this chapter.

6.1. T=CL ACTIVATION – ISO 14443-A

6.1.1. R-ATS

This function asks the currently selected ISO 14443-A card to enter T=CL level. The card answers with its ATS (Answer To Select).

Command

CMD	LEN	0	1
h81	h02	h10	CID

Value for CID could be either:

- hFF : CID not used (only one card active at a time),
- h00 to h0E : the card takes the specified CID.

Response OK

STA	LEN	0..xx-1
0	xx	ATS

6.1.2. PPS

This function asks the card to change its communication parameters.

Command

CMD	LEN	0	1	2	3
h81	h04	h11	<i>CID</i>	<i>DSI</i>	<i>DRI</i>

CID shall be the same as in *R-ATS*

DSI is the card-to-coupler baudrate

DRI is the reader-to-card baudrate

Allowed values for both *DSI* and *DRI* are:

- h00 : 106kbps (default),
- h01 : 212kbps,
- h02 : 424kbps,
- h03 : 847kbps.

NB: the host application must ensure that the specified parameters are actually supported by both the coupler and the card.

Response OK

STA	LEN
0	h00

6.2. T=CL ACTIVATION – ISO 14443-B

6.2.1. ATTRIB – Stay at 106kpbs

This function asks the currently selected ISO 14443-B card to enter T=CL level. The card answers with an ATTRIB RESPONSE (may be empty).

Command

CMD	LEN	0	1
h81	h02	h10	CID

Value for CID could be either:

- hFF : CID not used (only one card active at a time),
- h00 to h0E : the card takes the specified CID.

Response OK

STA	LEN	0..xx-1
0	xx	R-ATTRIB

6.2.2. ATTRIB + Set Baudrate

This function asks the currently selected ISO 14443-B card to enter T=CL level and changes its communication parameters. The card answers with an ATTRIB RESPONSE (may be empty).

Command

CMD	LEN	0	1	2	3
h81	h04	h10	CID	DSI	DRI

Value for CID could be either:

- hFF : CID not used (only one card active at a time),
- h00 to h0E : the card takes the specified CID.

DSI is the card-to-coupler baudrate

DRI is the reader-to-card baudrate

Allowed values for both DSI and DRI are:

- h00 : 106kbps (default),
- h01 : 212kbps,
- h02 : 424kbps,
- h03 : 847kbps.

NB: the host application must ensure that the specified parameters are actually supported by both the coupler and the card.

Response OK

STA	LEN	0..xx-1
0	xx	R-ATTRIB

6.3. T=CL APDU EXCHANGE

Command

CMD	LEN	0	1..xx-1
$h82$	xx	CID	Command to the card (C-APDU)

CID shall be the same as in R-ATS or ATTRIB

Response OK

STA	LEN	0..xx-1
0	xx	Response from the card (R-APDU)

6.4. T=CL DESELECT

This functions asks T=CL card to enter the HALT state.

Command

CMD	LEN	0	1
$h81$	$h02$	$h12$	CID

CID shall be the same as in R-ATS or ATTRIB

Response OK

STA	LEN
0	$h00$

7. ISO 15693

NB: not all couplers support ISO 15693 (and ICODE1). Before invoking one of the functions listed here, check that the connected coupler actually supports them, using the *Get Coupler Capabilities* command (3.1.2 and Table 1).

The functions listed in this chapter are available when the coupler's RF interface has been configured for either ISO 15693 'fast' mode or 'slow' mode (see § 3.2.1).

7.1. ISO 15693 CARD CONTROL

Please refer to ISO 15693-2 and -3 standard for details on this chapter.

7.1.1. Select Any

This function tries to select a compliant card.

Command

CMD	LEN
h40	h00

Response OK

CMD	LEN	0..7
h00	h08	UID

NB: for every standard ISO 15693 card, UID[0] = hE0

7.1.2. Select Again

This function tries to select again a card whose UID is already known.

Command *select last card again*

CMD	LEN
h44	h00

Command *select a specific card again*

CMD	LEN	0..7
h44	h08	UID

Response *OK*

STA	LEN
0	h00

Response *no answer*

STA	LEN
-1	h00

7.1.3. Halt

This command puts the currently selected card into the HALT state.

Command

CMD	LEN
h45	h00

Response *OK*

STA	LEN
0	h00

7.1.4. Get System Information

This function is **optional** according to the ISO 15693-3 standard. It may be available or not, depending on the card.

Command

CMD	LEN
h39	h00

Response *OK*

STA	LEN	1	1..8	9..xx-1
0	hXX	<i>Flags</i>	<i>UID</i>	<i>More Data</i>

The fields available in *More Data* are indicated by the *Flags*

- if (*Flags* & h01) → DSFID field present (1 byte),
- if (*Flags* & h02) → AFI field present (1 byte),
- if (*Flags* & h04) → Memory Size field present (2 bytes),
- if (*Flags* & h08) → IC reference field present (1 byte).

7.2. ISO 15693 FRAME EXCHANGE

7.2.1. Standard Frame Exchange

Command

CMD	LEN	0..1	2..3	4	5..6	6..xx-1
h98	hXX	<i>Send Len</i>	<i>Recv Len</i>	h01	<i>Timeout</i>	<i>Card's command</i>

Response OK

STA	LEN	0..1	2..xx-1
0	hXX	<i>Recv Len</i>	<i>Card's Response</i>

7.2.2. Advanced Frame Exchange

To be written

7.3. ISO 15693 READ

The functions listed here are **optional** according to the ISO 15693-3 standard. They may be available or not, depending on the card.

Please refer to the documentation of the card for details.

7.3.1. Read Block

Command

CMD	LEN	0
h46	h01	<i>Addr.</i>

Response OK

STA	LEN	0..xx-1
0	hXX	<i>Data</i>

7.3.2. Read Multiple Blocks

Command

CMD	LEN	0	1	2
h46	h03	<i>Addr.</i>	h00	<i>Count</i>

Response OK

STA	LEN	0..x-1
0	hXX	<i>Data</i>

7.3.3. Read Bytes

Command

CMD	LEN	0	1	2
h46	h03	<i>Addr.</i>	h01	<i>Count</i>

Response OK

STA	LEN	0..x-1
0	hXX	<i>Data</i>

Note: in this case LEN = *Count*

7.4. ISO 15693 WRITE AND LOCK

The functions listed here are **optional** according to the ISO 15693-3 standard. They may be available or not, depending on the card.

Please refer to the documentation of the card for details.

7.4.1. Write Block

Command

CMD	LEN	0	1..xx-1
h47	hXX	Addr.	Data

Response OK

STA	LEN
0	h00

7.4.2. Lock Block

Command

CMD	LEN	0
h59	h01	Addr.

Response OK

STA	LEN
0	h00

8. OTHER RF PROTOCOLS

8.1. INNOVATRON RADIO PROTOCOL (CALYPSO)

To be written

8.2. NXP ICODE1

To be written

9. AUTOMATIC CARD DISCOVERY

The *Automatic Card Discovery* functions offer an efficient way to detect cards on the RF interface, whatever their protocol.

9.1. FIND CARD

NB: this function was introduced in firmware 1.51 and is not available on earlier versions.

9.1.1. Single shot find

Command

CMD	LEN	0..1
h60	h02	<i>Protocols</i>

Protocols is a 16-bit field (MSB first). Set the bit(s) corresponding to the protocol(s) to be looked for, according to Table 2 on next page.

If *Protocols* is set to hFFFF, the coupler will use all the protocols it supports.

Response *no card in the field*

STA	LEN
-1	h00

Response *OK*

STA	LEN	0..1	2..xx-1
0	xx	<i>Card protocol</i>	<i>Card UID / PUPI</i>

Protocols is a 16-bit field (MSB first) where only one bit will be set, telling which kind of card has been found, according to Table 2 on next page.

Card UID / PUPI gives the protocol-level identifier of the card. Its size depends on the actual protocol (4, 7 or 10 for ISO 14443 type A, 4 for ISO 14443 type B, 8 for ISO 15693, etc). Please refer to the standards for details.

Invoke one of the *Get Card Information* commands (§ 9.2) to retrieve other card data.

Table 2: Card protocols

BIT	MASK	Meaning
0	h0001	ISO 14443-3 or -4, type A (including NXP Mifare family)
1	h0002	ISO 14443-3 or -4, type B
2	h0004	ISO 15693 (fast mode)
3	h0008	NXP ICODE1 (fast mode)
4	h0010	Inside Contactless PicoTag (including HID iClass)
5	h0020	ST MicroElectronics SRx
6	h0040	ASK CTS B
7	h0080	Innovatron radio protocol (early Calypso cards)
8	h0100	RFU
9	h0200	RFU
10	h0400	ISO 15693 (standard mode)
11	h0800	NXP ICODE1 (standard mode)
12	h1000	Innovision Topaz / Jewel
13	h2000	RFU
14	h4000	RFU
15	h8000	RFU

9.2. POLLING LOOPS

NB: those functions were introduced in firmware 1.54 and are not available on earlier versions.

9.2.1. Wait One Card

This function instructs the coupler to start waiting for a card, according to the specified protocol(s). The function exits when a card is found or a timeout occurs.

Command

CMD	LEN	0..1	2..3	4	5..6
$\text{h}60$	$\text{h}07$	<i>Protocols</i>	<i>Timeout (s)</i>	<i>Mode</i>	<i>Interval (ms)</i>

- *Protocols* is a 16-bit field (MSB first). Set the bit(s) corresponding to the protocol(s) to be looked for, according to *Table 2: Card protocols*.
- *Timeout (seconds)*: the coupler stops waiting when this timeout is expired. Set to hFFFF for an endless waiting.
- *Mode*: set to $\text{h}00$ for normal operation, or to $\text{h}01$ for power-saving mode.
This value could be omitted ($\text{LEN} \leq \text{h}04$) and defaults to $\text{h}00$.
- *Interval (milliseconds)*: this is the period between two consecutive lookup shots. Increasing this value will reduce coupler's average consumption (as the RF field is switched OFF in-between), but the coupler will be less "reactive". 250ms is the recommended value. Value hFFFF is forbidden.
This value could be omitted ($\text{LEN} \leq \text{h}05$) and defaults to 250 ($\text{h}00\text{FA}$).

This function is the only one (apart is sister-function *Wait Multiple Cards*) that returns two Responses:

1st Response *polling started*

STA	LEN
-20	$\text{h}00$

2nd Response *timeout expired or break, no card found*

STA	LEN
-30	$\text{h}00$

2nd Response *card found*

STA	LEN	0..1	2..xx-1
0	xx	<i>Card protocol</i>	<i>Card UID / PUPI</i>

Refer to § 9.1 for details.

NB: to cancel a running *Wait One Card* function, invoke the *Wait Cancel* function (§ 9.2.3) or one of the *Get Card Information* functions (§ 9.3). Any other function will fail with STA=-20.



When *Mode* = $\text{h}01$ (power-saving mode), the coupler **does not listen on its serial line** until the command is terminated.

Polling will terminate only when a card is found or when the timeout expires.

The only way to cancel a running command in this case is to reset the coupler.

9.2.2. Wait Multiple Cards

This function instructs the coupler to start waiting for a card, according to the specified protocol(s). The function exits only when a timeout occurs. When a card is found, a Response is sent, but the function keeps on looking for another card.

Command

CMD	LEN	0..1	2..3	4	5..6	7..8
$\text{h}60$	$\text{h}09$	<i>Protocols</i>	<i>Timeout (s)</i>	<i>Mode</i>	<i>Interval A (ms)</i>	<i>Interval P (ms)</i>

- *Protocols* is a 16-bit field (MSB first). Set the bit(s) corresponding to the protocol(s) to be looked for, according to *Table 2: Card protocols*.
- *Timeout* (seconds): the coupler stops waiting when this timeout is expired. Set to $\text{h}FFFF$ for an endless waiting.
- *Mode*: set to $\text{h}00$ for normal operation, or to $\text{h}01$ for power-saving mode.
- *Interval A* (milliseconds): this is the period between two consecutive lookup shots, when there's no card in the field. Increasing this value will reduce coupler's average consumption (as the RF field is switched OFF in-between), but the coupler will be less "reactive". 250ms is the recommended value. Value $\text{h}FFFF$ is forbidden.
- *Interval P* (milliseconds): this is the period between two consecutive lookup shots, when there's a card in the field. The "card found" Response will be repeated according to this period until the card is removed from the field. 50ms is the recommended value. Values $\text{h}0000$ and $\text{h}FFFF$ are forbidden.

This function may return more than two Responses.

1st Response *polling started*

STA	LEN
-20	$\text{h}00$

Xth Response *timeout expired or break, no card found*

STA	LEN
-30	$\text{h}00$

Xth Response *card found*

STA	LEN	0..1	2..xx-1
0	xx	Card protocol	Card UID / PUPI

Refer to § 9.1 for details.

NB: to cancel a running *Wait Multiple Cards* function, invoke the *Wait Cancel* function (§ 9.2.3) or one of the *Get Card Information* functions (§ 9.3). Any other function will fail with STA=-20.



When *Mode* = `h01` (power-saving mode), the coupler **does not listen on its serial line** until the command is terminated.

Polling will terminate only when the timeout expires.

The only way to cancel a running command in this case is to reset the coupler.

9.2.3. Wait Cancel

Invoke this function to cancel a running *Wait One Card* or *Wait Multiple Cards* function.

Command

CMD	LEN	0..1
<code>h60</code>	<code>h02</code>	<code>h0000</code>

Response *OK*

STA	LEN
0	<code>h00</code>

9.3. GET CARD INFORMATION

NB: these functions were introduced in firmware 1.51 and are not available on earlier versions.

9.3.1. Get Card UID/PUPI

Command

CMD	LEN	O
h61	h01	h01

Response OK

STA	LEN	0..1	2..xx-1
0	xx	Card protocol	Card UID / PUPI

Protocols is a 16-bit field (MSB first) where only one bit will be set, telling which kind of card has been found, according to Table 2 on next page.

Card UID / PUPI gives the protocol-level identifier of the card. Its size depends on the actual protocol (4, 7 or 10 for ISO 14443 type A, 4 for ISO 14443 type B, 8 for ISO 15693, etc). Please refer to the standards for details.

NB: this is exactly the same Response as in 9.1.1, 9.2.1 and 9.2.2.

NB: as a side effect, this function also cancels a running *Wait Card* or *Wait Multiple Cards* command.

9.3.2. Get Card Protocol Bytes

Command

CMD	LEN	O
h61	h01	h02

Response when card protocol = ISO 14443 type A

STA	LEN	0..1	2
0	h03	ATQ	SAK

Response when card protocol = ISO 14443 type B

STA	LEN	0..10
0	h0B	ATQ

Response when card protocol = Innovatron

STA	LEN	0..xx-1
0	xx	REPGEN

Response other card protocols

STA	LEN
0	h00

NB: as a side effect, this function also cancels a running *Wait Card* or *Wait Multiple Cards* command.

10. STATUS AND ERROR CODES

The *symbolic names* are the one used in SpringProx API.

10.1. SUCCESS AND SPECIAL STATUS

STA	Symbolic name	Meaning
0	MI_OK	Success
-20	MI_POLLING	Polling mode pending
-30	MI_QUIT	Polling terminated (timeout or break ⁷)
-128	MI_TIME_EXTENSION	Coupler is still processing the Command

10.2. ERRORS IN RF COMMUNICATION OR PROTOCOL

STA	Symbolic name	Meaning
-1	MI_NOTAGERR	No answer (no card / card is mute)
-2	MI_CRCERR	Invalid CRC in card's response
-3	MI_EMPTY	No frame received (NFC mode)
-5	MI_PARITYERR	Invalid parity bit(s) in card's response
-7	MI_CASCLEVEX	Too many anti-collision loops
-8	MI_SERNRERR	Wrong LRC in card's serial number
-11	MI_BITCOUNTRERR	Wrong number of bits in card's answer
-12	MI_BYTECOUNTRERR	Wrong number of bytes in card's answer
-21	MI_FRAMINGERR	Invalid framing in card's response
-24	MI_COLLERR	A collision has occurred
-28	MI_NOBITWISEANTICOLL	More than one card found, but at least one does not support anti-collision
-29	MI_EXTERNAL_FIELD	An external RF field has been detected
-31	MI_CODING_ERR	Bogus status in card's response

10.3. ERRORS REPORTED BY THE CARD

STA	Symbolic name	Meaning
-4	MI_AUTHERR	Authentication failed or access denied
-6	MI_CODEERR	NACK or status indicating error
-9	MI_LOCKED	Card or block locked
-10	MI_NOTAUTHERR	Authentication must be performed first
-13	MI_VALUERR	Counter is invalid
-14	MI_TRANSERR	Transaction error
-15	MI_WRITEERR	Write failed
-16	MI_INCRERR	Counter increase failed
-17	MI_DECRERR	Counter decrease failed
-18	MI_READERR	Read failed
-22	MI_ACCESSERR	Access error (bad address or denied)
-32	MI_CUSTERR	Vendor specific error
-33	MI_CMDSUPERR	Command not supported
-34	MI_CMDFMTRERR	Format of command invalid
-35	MI_CMDOPTERR	Option(s) of command invalid
-36	MI_OTHERERR	Other card error

⁷ Polling is terminated by a serial BREAK (more than 12 ETU at level 0), when receiving 15 empty bytes (0x00) or by the ASCII {ESCAPE} character (0x1B)

10.4. ERRORS AT T=CL LEVEL

STA	Symbolic name	Meaning
-71	MI_CID_NOT_ACTIVE	No active card with this CID
-75	MI_BAD_ATS_LENGTH	Length error in card's ATS
-76	MI_ATTRIB_ERROR	Error in card's response to ATTRIB
-77	MI_BAD_ATS_FORMAT	Format error in card's ATS
-78	MI_TCL_PROTOCOL	Protocol error in card's response
-87	MI_BAD_PPS_FORMAT	Format error in card's PPS response
-88	MI_PPS_ERROR	Other error in card's PPS response
-93	MI_CID_ALREADY_ACTIVE	A card is already active with this CID

10.5. OTHER ERRORS

STA	Symbolic name	Meaning
-19	MI_OVFLERR	RC FIFO overflow
-23	MI_UNKNOWN_COMMAND	Unknown RC command
-25	MI_COMMAND_FAILED	Command execution failed
-26	MI_INTERFACEERR	Hardware error
-27	MI_ACCESSTIMEOUT	RC timeout
-59	MI_WRONG_MODE	Command not available in this mode
-60	MI_WRONG_PARAMETER	Wrong parameter for the command
-100	MI_UNKNOWN_FUNCTION	Command not supported by the coupler
-112	MI_BUFFER_OVERFLOW	Internal buffer overflow
-125	MI_WRONG_LENGTH	Wrong data length for the command

DISCLAIMER

This document is provided for informational purposes only and shall not be construed as a commercial offer, a license, an advisory, fiduciary or professional relationship between PRO ACTIVE and you. No information provided in this document shall be considered a substitute for your independent investigation.

The information provided in this document may be related to products or services that are not available in your country.

This document is provided "as is" and without warranty of any kind to the extent allowed by the applicable law. While PRO ACTIVE will use reasonable efforts to provide reliable information, we don't warrant that this document is free of inaccuracies, errors and/or omissions, or that its content is appropriate for your particular use or up to date. PRO ACTIVE reserves the right to change the information at any time without notice.

PRO ACTIVE does not warrant any results derived from the use of the products described in this document. PRO ACTIVE will not be liable for any indirect, consequential or incidental damages, including but not limited to lost profits or revenues, business interruption, loss of data arising out of or in connection with the use, inability to use or reliance on any product (either hardware or software) described in this document.

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products may result in personal injury. PRO ACTIVE customers using or selling these products for use in such applications do so on their own risk and agree to fully indemnify PRO ACTIVE for any damages resulting from such improper use or sale.

COPYRIGHT NOTICE

All information in this document is either public information or is the intellectual property of PRO ACTIVE and/or its suppliers or partners.

You are free to view and print this document for your own use only. Those rights granted to you constitute a license and not a transfer of title: you may not remove this copyright notice nor the proprietary notices contained in this document, and you are not allowed to publish or reproduce this document, either on the web or by any mean, without written permission of PRO ACTIVE.

Copyright © PRO ACTIVE SAS 2010, all rights reserved.

EDITOR'S INFORMATION

PRO ACTIVE SAS company with a capital of 227 000 €
RCS EVRY B 429 665 482
Parc Gutenberg, 13 voie La Cardon
91120 Palaiseau – France